POLITECHNIKA ŚLĄSKA BIBLIOTEKA GŁÓWNA GLIWICE
P. 1877/75

6 1975

informatyka

SPIS TREŚCI NR 6/75

Zautomatyzowany system obsługi informacyjnej VI Halowych Mistrzostw Europy w Lekkiej Atletyce — *Włodzimierz Marciński* 1

Organizacja systemów automatyzacji projektowania — *Krzysztof Urbaniec* 5

Organizacja i wyszukiwanie informacji za pomocą metod transformacyjnych — *Wacław Iszkowski* 8

Porównanie parametrów i możliwości komputerowych systemów graficznych — *Krzysztof Tannenberg* 12

Rachunek kosztów w świetle badań ankietowych — *Bogumił Stachura* 16

SOCRATE — system zarządzania bazami danych — *Helena Dryzek* 19

SYSFAP — zintegrowany system automatycznie programowanych zbiorów danych — *Stefan Grodzki, Maciej Winiarski* 22

Programowanie systemowe — nowopowstająca dziedzina nauki — tłum. i oprac. *Marek Misala* 26

INFORMATYKA NA UCZELNIACH — Wiadomości różne 15

WIADOMOŚCI PKAPI — Prezydium PKAPI w Rzeszowie 25

TRYBUNA CZYTELNIKA — O markietingu informatycznym słów kilka — *Kazimierz Martyniak* 31

Z KRAJU — Polski komputer R-32 33

Wiadomości różne 34

ZE ŚWIATA — Zastosowanie metod matematycznych i informatyki w geologii krajów RWPG — *Maciej K. Radecki* 35

IBM oskarżony o naruszenie amerykańskich ustaw antytrustowych — *Jerzy Miłek* 36

Wiadomości różne 37

ZE ZJEDNOCZENIA INFORMATYKI — Problemy oprogramowania i zadania, jakie w związku z jego produkcją stoją przed Zjednoczeniem Informatyki 39

TERMINOLOGIA — Programowanie — opis danych w języku programowania — *Władysław Klepacz* 42

SZTUKA KOMPUTEROWA — Komputery w muzyce — *Marek Hołyński* 43

PROBLEMATYKA BAZY DANYCH — Koncepcja Uniwersalnego Systemu Zarządzania Bazą Danych dla komputerów JS — *Władysław Bogucki, Witold Staniszkis* 45

PRZEGLĄD WYDAWNICTW — Recenzja książki Janusza Hawryluka: Maszyna cyfrowa narzędzie człowieka współczesnego — *Józef Czajkowski* 47

Bibliografia wydawnictw polskich z dziedziny informatyki — oprac. *J. Klamborowski* 48, III i IV skrz.

WYDAWNICTWA CZASOPISM TECHNICZNYCH NOT
Warszawa
Czackiego 3/5

KOLEGIUM REDAKCYJNE

Redaktor Naczelny prof. dr hab. Leon ŁUKASZEWICZ

mgr Krystyn BERNATOWICZ, prof. dr hab. inż. Konrad FIAŁKOWSKI (zast. redaktora naczelnego), doc. dr inż. Zbigniew GACKOWSKI, mgr inż. Marek HOŁYŃSKI, Władysław KLEPACZ, doc. dr hab. Antoni MAZURKIEWICZ,

Sekretarz Redakcji mgr Krystyna WROŃSKA — Red. tech. Józef DUSZA

RADA PROGRAMOWA

Mgr inż. Antoni BOSSOWSKI, doc. dr inż. Jan FELICKI, doc. dr inż. Zbigniew GACKOWSKI, doc. dr inż. Aleksander GOLINOWSKI, dr hab. Andrzej GRZYWAK, doc. dr hab. inż. Roman KULESZA, prof. dr hab. Leon ŁUKASZEWICZ, prof. dr hab. inż. Stanisław PASZKOWSKI (wiceprzewodniczący), prof. dr Tadeusz PECHE, mgr inż. Bronisław PIWOWAR, dr inż. Andrzej PLĄSKOWSKI, mgr inż. Tadeusz PODGÓRSKI (wiceprzewodniczący), prof. dr inż. Jerzy SEIDLER, prof. dr inż. Andrzej STRASZAK (przewodniczący), doc. Jerzy TRYBULSKI, dr Tadeusz WALCZAK, prof. dr inż. Stefan WĘGRZYN, dr inż. Jan Z. ŻYDOWO

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pokój 331, tel. 27-71-40 lub centrala 28-82-61 w. 285, dyżury redakcji 10,00—13,00

Zakład Kolportażu WCT NOT, Warszawa, ul. Mazowiecka 12

Zakł. Graf. „Tamka". Z. 2. Zam. 228. Papier druk. sat IV kl. 70 g 61 + 86. Obj. 6,75 ark. druk. Nakład 5600. B-88.

Cena egzemplarza zł 15.— INDEKS 36210/36124 Prenumerata roczna zł 180.—

Informatyka

dawniej Maszyny Matematyczne

zastosowania w gospodarce, technice i nauce

Nr 6

MIESIĘCZNIK

1975

ROK XI

Czerwiec

ORGAN MINISTERSTWA NAUKI, SZKOLNICTWA WYŻSZEGO I TECHNIKI I POLSKIEGO KOMITETU AUTOMATYCZNEGO PRZETWARZANIA INFORMACJI NACZELNEJ ORGANIZACJI TECHNICZNEJ

POLITECHNIKA ŚLĄSKA BIBLIOTEKA GŁÓWNA GLIWICE

I. 1877/75

WŁODZIMIERZ MARCIŃSKI
Ośrodek Badawczo-Rozwojowy Urządzeń Informatyki MERA
Warszawa

681.322-181.4.004.14:796.4

Zautomatyzowany system obsługi informacyjnej VI Halowych Mistrzostw Europy w Lekkiej Atletyce

Za postępującym rozwojem sportu zdecydowanie w tyle pozostawała organizacja zawodów sportowych, w której poza elektronicznym pomiarem czasu, nie zaszły żadne zmiany od kilkudziesięciu lat. Czynnikiem, który jest w stanie gruntownie zreorganizować i uprawnić najważniejszy element organizacji imprezy sportowej — obieg informacji, jest elektroniczna technika obliczeniowa.
Pierwsze zastosowanie maszyn cyfrowych podczas zawodów sportowych ściśle wiążą się z reklamą urządzeń. Do dziś jeszcze z wyjątkiem Olimpiady w Monachium, firmy komputerowe takie jak IBM, HONEYWELL, SINGER, NIXDORF, PHILIPS-UNIDATA, w swych „sportowych" zastosowaniach widzą przede wszystkim korzyści reklamowe.
Odbija się to na realizowanych systemach, które nie obejmują tym samym kompleksu zagadnień związanych z obsługą informacyjną zawodów. Odrębnym typem zastosowań był, oparty na komputerach SIEMENS 4004, System Banku Danych GOLYM, wykorzystany podczas igrzysk olimpijskich w Monachium.
W Polsce pierwszym kompleksowym wykorzystaniem ETO na zawodach sportowych, było użycie minikomputera K-202 do sterowania organizacją zawodów, sterowania tablicą świetlną, obliczeń typu statystycznego i informacji prasowej podczas Mistrzostw Świata w Zapasach (Katowice, kwiecień 1972 r.).
W czasie Halowych Lekkoatletycznych Mistrzostw Europy w Katowicach wykorzystano nowy system opracowany na MERĘ 305.

Omówiono system obsługi informacyjnej zrealizowany na minikomputerze MERA 305. Przedstawiono zadania systemu oraz konfigurację zastosowanego sprzętu.

System Obsługi Informatycznej VI Halowych Mistrzostw Europy w Lekkiej Atletyce

Po wstępnym rozpoznaniu zadań, jakie stały przed systemem, zdecydowano, że rozwiązanie ich jest możliwe za pomocą minikomputera MERA 305 w jego standardowej konfiguracji z jednostką pamięci dyskowej oraz drukarką znakowo-mozaikową DZM 180. Wytypowanie tej właśnie maszyny pociągało za sobą pewne ryzyko. Projektowany system miał być pierwszym zastosowaniem tego modelu serii minikomputerów MERA 300. Okres, w którym miało nastąpić uruchomienie i sprawdzenie systemu składającego się z 22 programów nie przekraczał półtora miesiąca. Ryzyko to zostało podjęte.

Podstawowymi zadaniami systemu było:

- natychmiastowe reagowanie na zmiany, które mogłoby zajść przed oraz w trakcie zawodów (np. skreślenia z list startowych)
- bieżące informowanie widzów zgromadzonych na hali o aktualnie uzyskiwanych wynikach oraz składach poszczególnych konkurencji
- dostarczanie oficjalnych komunikatów dziennikarzom i sprawozdawcom radia i telewizji, w możliwie najkrótszym czasie po zakończeniu każdej konkurencji
- sporządzanie dokumentacji pierwotnej mistrzostw (protokóły konkurencji), list startowych oraz zbiorczego komunikatu końcowego.

W skład środków technicznych omawianego systemu poza dwoma minikomputerami MERA 305 wchodziły: dwie kopiarki kserograficzne RANK XEROX 3600 oraz tablica świetlna produkcji krajowej zainstalowana w hali. Tablica ta stanowi najistotniejsze ograniczenie systemu.

Mgr WŁODZIMIERZ MARCIŃSKI ukończył kierunek Organizacji Przetwarzania Danych SGPiS (1972). Na studiach oraz w Instytucie Maszyn Matematycznych, gdzie pracował jako asystent zajmował się problemami związanymi z wykorzystaniem komputerów klawiaturowych. Obecnie pracuje nad tymi samymi zagadnieniami w Ośrodku Badawczo-Rozwojowym Urządzeń Informatyki MERA. Specjalizuje się w zastosowaniach maszyn cyfrowych w sporcie. Jest autorem opisanego w artykule zastosowania minikomputera MERA 305 w VI Halowych Mistrzostwach Europy w Lekkiej Atletyce.

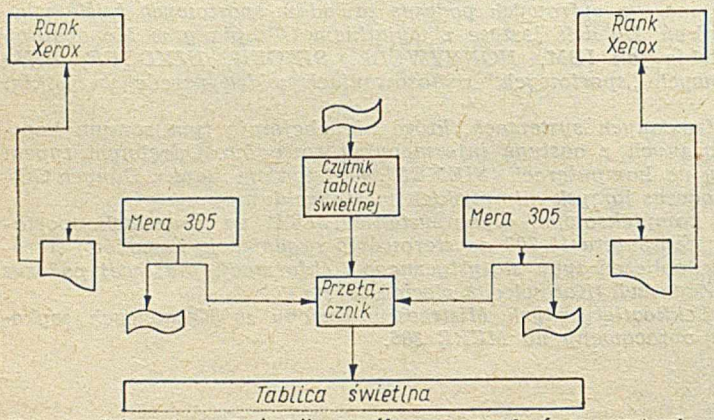
Wprowadzane na nią informacje musiały mieścić się w polu 12 × 40 znaków, które podawane mogły być jedynie w kodzie telegraficznym M-2. Zadaniem kopiarek było powielenie wydruków maszynowych z maksymalną prędkością 60 odbitek na minutę.

Zabezpieczeniem na wypadek jakiejkolwiek awarii sprzętu było zdublowanie wszystkich podstawowych urządzeń technicznych zaangażowanych w obsłudze informatycznej — dwa zestawy MERA 305, dwa czytniki tablicy świetlnej, dwa kserografy.

Takie zabezpieczenie nie było przesadą, ponieważ każdy nieco dłuższy przestój mógłby zburzyć koncepcję organizacyjną mistrzostw. Zgodnie z założeniami oba minikomputery miały pracować równolegle, utrzymując nieprzerwanie identyczny obieg informacji.

Pomiędzy próbą generalną systemu, która odbyła się podczas Halowych Lekkoatletycznych Mistrzostw Polski w Katowicach w dniach 22—23.02.1975 r. a imprezą główną t.zn. Mistrzostwami Europy, które odbyły się w Katowickiej Hali Widowiskowo-Sportowej w dniach 8—9 marca 1975 roku, istniała duża różnica w metodzie współpracy minikomputerów z tablicą świetlną. Na pierwszej ze wspomnianych imprez, wszystkie informacje wchodzące na tablicę świetlną przenoszone były za pośrednictwem taśm dziurkowanych. Takie rozwiązanie obarczone było prawdopodobieństwem błędów zarówno przy perforowaniu jak i wczytywaniu taśm. Po dokładnym zapoznaniu się z istniejącymi warunkami, pracownicy OBRUI — inżynierowie W. Brzeski i W. Zajdel — w przeciągu kilkunastu dni zaprojektowali, wykonali i zainstalowali urządzenie pozwalające na bezpośrednie połączenie obu MER z tablicą świetlną. Konstrukcja miała przełącznik umożliwiający alternatywne wprowadzanie informacji z jednego z minikomputerów lub z czytników tablicy (rys. 1).

Niezależnie od zainstalowanego połączenia wszystkie informacje przenoszone bezpośrednio, wyprowadzane były na taśmę dziurkowaną. Taśmy te wykorzystywane były do



Rys. 1. Schemat organizacji współpracy urządzeń w systemie MELA

powtórzeń wyników przy akcie dekoracji zawodników.
Opracowany system obsługi informatycznej nazwany roboczo MELA (MISTRZOSTWA EUROPY W LEKKIEJ ATLETYCE) opierał się na dwóch podstawowych zbiorach założonych na dysku wymiennym maszyny, tj.: zbiorze nazw stałych i informacji o zgłoszonych zawodnikach.
Do nazw stałych systemu zaliczone zostały wszystkie wyrażenia używane w komunikatach i opracowaniu końcowym.

Każde z tych wyrażeń miało reprezentację w trzech językach: polskim, angielskim i francuskim. Temu zbiorowi przyporządkowany został dwupoziomowy system kodów numerycznych.

Na przykład na pierwszym poziomie konkurencja „60 metrów kobiet" miała kod 01 a „skok o tyczce" — kod 19, natomiast na drugim poziomie „przedbieg 2" — kod 02 a „półfinał 1" — kod 09.

Zbiór informacji o zgłoszonych zawodnikach składał się z zapisów, z których każdy obejmował pełen zestaw niezbędnych danych dotyczących jednego uczestnika mistrzostw.

Poza skrótem imienia i nazwiskiem zawodnika w skład zapisu wchodziły: numer startowy zawodnika trzyliterowy

symbol państwa, numeryczny symbol państwa, numeryczne symbole konkurencji, do których dany zawodnik został zgłoszony oraz najlepsze jego wyniki halowe w zgłoszonych konkurencjach. Z uwagi na konieczność szybkiego, bezpośredniego dostępu do wszystkich danych zbioru — zapis dotyczący jednego zawodnika rozlokowany był w jednym sektorze dysku. Dostęp do informacji o zawodniku odbywał się poprzez jego numer startowy, który odwzorowywany był w adres dyskowy.

Ze zbioru danych o uczestnikach mistrzostw korzystały prawie wszystkie programy systemu.

Przyjmując jako kryterium czas rozgrywania mistrzostw całość programów systemu MELA podzielić można na trzy grupy: programów realizowanych przed, w trakcie i po zawodach.

Podstawową czynnością możliwą do wykonania na kilka dni przed zawodami jest założenie zbiorów podstawowych systemu. Zbiór nazw stałych zakładany był jednarazowo, natomiast zbiór danych o zawodnikach był zakładany przez kilka dni, sukcesywnie w trakcie napływania zgłoszeń z różnych państw.

Po zamknięciu zgłoszeń i wprowadzeniu danych o wszystkich zawodnikach — za pomocą programów systemu MELA — sporządzone zostały listy zgłoszeń w układzie państw oraz listy zgłoszeń w układzie konkurencji.
Program realizujący pierwsze ze wspomnianych list wymagał podania symbolu państwa oraz rozdziału numerów startowych przydzielonych jego reprezentantom. W ramach każdego państwa prowadzona była odrębna numeracja zawodników umożliwiająca szybkie sprawdzenie stanu ilościowego ze zgłoszeniem.

Sporządzenie list zgłoszeń w układzie konkurencji wymagało podania jedynie symbolu konkurencji. Automatycznie wyszukiwani byli wszyscy zgłoszeni do niej zawodnicy. Poza numerem skrótowym, skrótem imienia, nazwiskiem oraz symbolem państwa na wydruku wypisywany był również najlepszy wynik halowy danego uczestnika. Listy zgłoszeń w układzie konkurencji stanowiły podstawowy dokument, na podstawie którego na konferencji technicznej mistrzostw losowane były przedbiegi i tory dla konkurencji biegowych oraz kolejności startu dla konkurencji technicznych.

Lista zgłoszeń w układzie państw oraz listy zgłoszeń w układzie konkurencji zostały przedrukowane i znalazły się w biuletynie informacyjnym mistrzostw.

W systemie MELA zaprojektowano również specjalne algorytmy rozlosowujące przedbiegi i przydzielające tory zawodnikom. Z uwagi na to, że automatyzacja wspomnianych czynności musiałaby być zatwierdzona przez Europejską Federację Lekkiej Atletyki, co byłoby możliwe dopiero na dwa dni przed mistrzostwami, zrezygnowano z ich zaprogramowania.

Na dwie godziny przed oficjalnym otwarciem mistrzostw odbywała się konferencja techniczna, na której poza powyżej wspomnianymi rozlosowaniami dokonano ostatecznych skreśleń i uzupełnień listy startowej.

Zadaniem systemu MELA było teraz ostateczne zaktualizowanie zbioru informacji o zawodnikach, przygotowanie komunikatu wstępnego zawierającego listy startowe w od-

powiednich układach, wydrukowanie protokółów dla wszystkich konkurencji mistrzostw oraz przygotowanie taśm dziurkowanych z zapowiedziami startów na tablicę świetlną.

Realizacja w tak krótkim czasie wszystkich wymienionych prac metodą ręczną wymagałaby zatrudnienia kilkunastu maszynistek i pracowników sekretariatu. Ponadto precyzja wykonania byłaby daleka od uzyskanej za pomocą minikomputera.

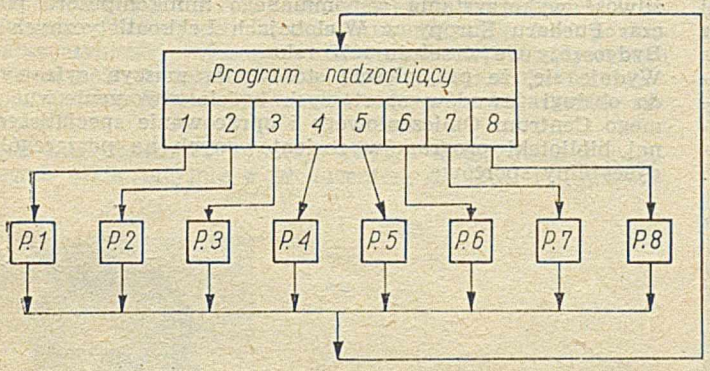
Przy sporządzaniu protokółów sędziowskich bardzo ściśle powiązano ze sobą pracę MERY 305 i kserografu. Na drukarce minikomputera powstawał czysty wydruk zgłoszonych do danego biegu czy konkurencji technicznej zawodników, natomiast na wydruk ten — za pomocą kserografu i specjalnych klisz — nakładano nadruk schematu formularza obejmującego rubryki na wyniki i miejsce uzyskane przez poszczególnych zawodników w danej konkurencji. W ten sposób zautomatyzowane zostało powstawanie dokumentacji pierwotnej zawodów.

Równocześnie z wypisywaniem list startowych konkurencji eliminacyjnych, półfinałowych i finałowych, tworzone były taśmy dziurkowane w kodzie M-2 stanowiące zapowiedzi startów. Współpraca z tablicą świetlną zainstalowaną w Katowickiej Hali Widowiskowo-Sportowej w najistotniejszy sposób ograniczała sprawność eksploatacyjną opracowanego systemu. Wszystkie programy systemu napisane zostały w języku KOMPUTER BIUROWY, którego naturalnym kodem wyjściowym jest kod ISO-7. Wspomniana konieczność dostarczenia do tablicy informacji w kodzie M-2 wymagała użycia dość skomplikowanego dekodera. Proces dekodowania został rozwiązany za pomocą specjalnego pola na dysku, stanowiącego obraz tablicy świetlnej. Każdy program na którego wyjściu znajdowały się informacje wyprowadzane na tablicę świetlną, w pierwszym kroku zapisywał na dysku jej obraz, który w następnym kroku był dekodowany.

Ostatnią czynnością poprzedzającą przetwarzanie w czasie zawodów było zakładanie pól wyników konkurencji technicznych.

Program realizujący tę czynność rezerwował dla każdej z wspomnianych konkurencji odpowiedni obszar na dysku. Zapisywane w nim były numery startowe zawodników zgłoszonych do danej konkurencji. W trakcie zawodów pole to stanowiło zbiór wyników uzyskiwanych w kolejnych seriach.

W trakcie trwania Mistrzostw, system MELA miał na celu:
— tworzenie komunikatów-ulotek z wynikami poszczególnych przedbiegów, półfinałów i finałów
— wyprowadzenie na tablicę świetlną wyników aktualnie rozegranych konkurencji
— zbieranie i przechowywanie wyników uzyskiwanych w konkurencjach technicznych, a na ich podstawie prowadzenie aktualnej klasyfikacji
— tworzenie protokółów dla półfinałów i finałów
— tworzenie na taśmach zapowiedzi dla półfinałów i finałów.



Rys. 2. Zamknięty obwód programów systemu MELA

Funkcję nadzorczą wypełniał specjalny program przekazujący sterowanie do odpowiednich segmentów realizujących wybraną czynność (rys. 2). Wszystkie główne funkcje systemu zostały ponumerowane od 1 do 8 i tak przykładowo przejście do tworzenia komunikatów-ulotek wymagało wprowadzenia z klawiatury cyfry 2 do dekodera kodu M-2 cyfry 5 itd. Każdy specjalizowany program kończył się ponownym wywołaniem programu nadzorującego.

Dla konkurencji technicznych, w których uzyskany wynik jednoznacznie decydował o zajmowanym miejscu (rzuty, skoki w dal itp.), przeznaczony był zespół powiązanych wewnętrznie programów. Napisane one były w taki sposób, aby korzystanie z nich mogło następować w trybie konwersacyjnym.

Przy wprowadzaniu wyników program pytał o symbol konkurencji i numer kolejki, przy analizowaniu o procedurę odsyłania, sortowania, wydrukowania czy poprawiania błędów.

Algorytm wykorzystany przy konkurencjach technicznych zakładał wprowadzanie wyników kolejki, w trakcie którego następowało porównywanie i wybieranie dwu najlepszych rezultatów danego zawodnika. Kolejność po zakończonej kolejce ustalana była drogą sortowania, w przypadku uzyskania przez kilku zawodników tego samego wyniku, o zajętym miejscu decydował drugi w kolejności wynik.

Aktualna klasyfikacja mogła być wprowadzana natychmiast bezpośrednio z minikomputera na tablicę świetlną. Pełny wydruk ostatecznej kolejności dokonywany był po trzeciej i ostatniej serii danej konkurencji. Poza najlepszymi wynikami zawierał on również wszystkie rezultaty osiągnięte w konkursie. Zespół programów obsługujących konkurencje techniczne był wywoływany i wracał do programu nadzorującego.

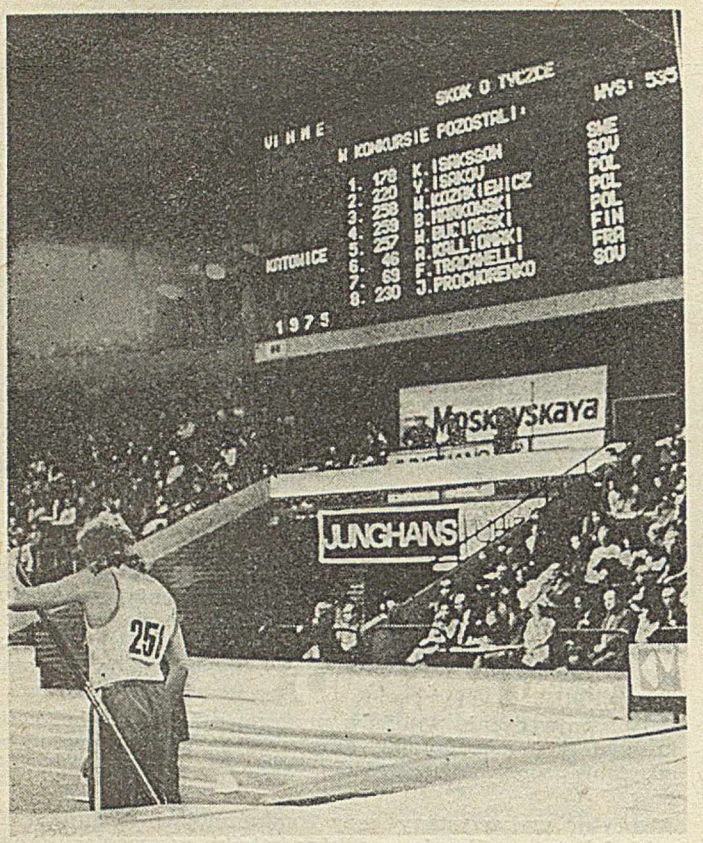
W ten sposób, w trakcie trwania mistrzostw, system przetwarzania stał się jak widać na rys. 2 zamkniętym obwodem programów.

Podstawą do sporządzenia wszystkich wydruków czy obrazów tablicy świetlnej były oficjalne protokóły sędziowskie. Dla konkurencji biegowych przenoszone były z nich do maszyny numery startowe, uzyskane czasy oraz ustalone kolejności miejsc. Z uwagi na to, że o zajętej lokacie nie decydował uzyskany czas, algorytm programu przyjmował zewnętrzne podawanie kolejności. Nieco inna była sytuacja w przypadku konkurencji technicznych. Z protokółów sędziowskich przenoszone były tylko numery startowe i wyniki.

Wprowadzenie wszystkich informacji na bieżąco z klawiatury wymagało dużej uwagi i skupienia. Aby wyłapać jak najwięcej błędów możliwych do popełnienia w takim trybie pracy, programy systemu MELA zaopatrzone były w elementy kontrolujące ręczne wprowadzanie. Sygnalizowana była próba wywołania programu o niezadeklarowanym numerze. W konkurencjach biegowych, próba wprowadzenia numeru startowego przekraczającego dopuszczalny przedział (1—600), powodowała odpowiednią sygnalizację oraz skierowanie sterowania na ponowne wprowadzenie.

W konkurencjach technicznych sygnalizowane były błędy polegające na wprowadzeniu numeru startowego zawodnika, który nie był zgłoszony do danej konkurencji, oraz wprowadzeniu wyniku przekraczającego praktycznie ludzkie możliwości. System nie przyjmował na przykład w rzucie kulą wyników powyżej 30 metrów.

Bezpośrednio po zakończeniu każdego dnia zawodów należało sporządzić zbiorczy komunikat zawierający wszystkie uzyskane wyniki. Wydawnictwo to różniło się od zbioru komunikatów-ulotek tym, że na jednej jego stronie znajdowały się wyniki z kilku przedbiegów lub finałów.



Następujące po sobie wyniki poukładane były zgodnie z planem minutowym dnia. Potrzeba takiej formy wydawnictwa końcowego mistrzostw wypłynęła ze względów estetycznych i oszczędnościowych. Według obliczeń organizatorów mistrzostw, powielenie skumulowanego komunikatu końcowego pochłonęło trzykrotnie mniej papieru kserograficznego w porównaniu do wydawnictwa składającego się ze wspomnianych komunikatów-ulotek. Oszczędność ta nabiera istotnych wymiarów gdy doda się, że wszystkie komunikaty powielane były w 600 egzemplarzach, a w trakcie mistrzostw rozegranych zostało 58 przedbiegów, półfinałów i finałów.

Organizacja obiegu informacji Mistrzostw

Wprowadzenie nowoczesnych środków przetwarzania danych do obsługi Halowych Mistrzostw Europy w Lekkiej Atletyce zmieniło w sposób istotny obieg informacji oraz metody ich rozpowszechniania. Na kilka minut przed rozpoczęciem każdej konkurencji, za pomocą czytnika wprowadzana była na tablicę świetlną zapowiedź startu, obejmująca nazwę, rodzaj konkurencji, numery startowe nazwiska i symbole państw reprezentowanych przez poszczególnych zawodników.

Proces informacyjny konkurencji biegowych rozpoczynali pracownicy firmy JUNGHANS, którzy w kilkadziesiąt sekund po zakończeniu biegu nanosili czasy do protokółu sędziowskiego. Zajęte miejsca wpisywane były przez specjalną komisję sędziowską. Opisany w ten sposób protokół schodził do centrum przetwarzania informacji, przeciętnie po upływie 1,5—2 minut po zakończeniu konkurencji.

Sporządzanie przy użyciu jednego z minikomputerów, MERA 305 wydruku komunikatu-ulotki z jednoczesnym zapisaniem obrazu tablicy świetlnej w pamięci dyskowej, trwało przeciętnie od 30 do 60 sekund. Czas ten zależał od liczby startujących zawodników. Wydruk wyników był zatem gotowy do powielenia około 3 minuty po zakończeniu biegu. W tym samym czasie co powielanie, możliwe były do zrealizowania dwie alternatywy:

— wyprowadzenie zgromadzonych informacji bezpośrednio na tablicę świetlną

— wyprowadzenie zgromadzonych informacji na taśmę dziurkowaną.

O tym czy informacje wyprowadzać na tablicę, czy też taśmę decydowali organizatorzy Mistrzostw, którzy niejednokrotnie z niewiadomych powodów wstrzymywali proces informowania. Wyświetlenie pełnego zapisu na tablicy świetlnej trwało około 1 minuty 10 sekund.

Podawanie do wiadomości aktualnej sytuacji w konkurencjach technicznych było możliwe po około 2 minutach po zakończeniu każdej kolejki.

Powielone komunikaty-ulotki roznoszone były za pośrednictwem gońców do zgromadzonych dziennikarzy i sprawozdawców radiowo-telewizyjnych oraz kierownictwa uczestniczących ekip.

Zbiorcze komunikaty końcowe dostarczone zostały wszystkim zainteresowanym w dwie godziny po zakończeniu zawodów.

Podstawowymi zaletami systemu obsługi informatycznej VI Halowych Mistrzostw Europy w Lekkiej Atletyce było:

— ujęcie w jedną organizacyjnie całość oraz zautomatyzowanie ogółu prac związanych z obsługą informacyjną imprezy, począwszy od tworzenia dokumentacji pierwotnej, a skończywszy na zbiorczym komunikacie końcowym

— uniknięcie błędów w pisowni nazwisk i nazw, co jest zmorą wszystkich międzynarodowych zawodów sportowych

— umożliwienie podawania uzyskanych wyników do powszechnej wiadomości po upływie około 4—5 minut po zakończeniu konkurencji (podczas poprzednich Mistrzostw czas ten wynosił około 10 minut).

Omówiony system spotkał się z dużym zainteresowaniem widzów, sportowców i dziennikarzy, którzy z uznaniem wyrażali się o jego funkcjonowaniu. Mistrzostwa stanowiły pokaz praktycznego działania polskiego sprzętu informatycznego, który zdał swój trudny egzamin.

Należy sądzić, że działacze i władze sportowe szerzej wykorzystają korzyści płynące z zastosowania elektronicznej techniki obliczeniowej.

Planowany jest udział minikomputera MERA-305 w pracach Biura Obliczeń IV Centralnej Spartakiady Młodzieży w Białymstoku w lipcu 1975 roku. Rozpatrywana jest możliwość wykorzystania wspomnianego minikomputera podczas Pucharu Europy w Wielobojach Lekkoatletycznych w Bydgoszczy we wrześniu 1975 roku.

Wydaje się, że przyszłością stosowania maszyn cyfrowych do obsługi zawodów sportowych będzie stworzenie ruchomego Centrum Obliczeniowego i opracowanie specjalistycznej biblioteki programów zorientowanych na poszczególne dyscypliny sportowe.

KRZYSZTOF URBANIEC
Politechnika Warszawska

681.322.6.004.14:62.001.13

Oprogramowanie systemów automatyzacji projektowania

Omówiono tradycyjne formy oprogramowania stosowanego w pracach projektowych oraz koncepcje oprogramowania systemów automatyzacji projektowania. Przedstawiono warunki efektywnej realizacji systemów oraz wnioski z zagranicznych doświadczeń w tej dziedzinie.

W biurach projektowych i konstrukcyjnych coraz częściej wykorzystuje się komputer wśród innych narzędzi pracy. Odbywa się to z rozmaitymi trudnościami — w znacznej mierze dlatego, że wprowadzaniu nowoczesnych narzędzi nie towarzyszą odpowiednie zmiany metod i organizacji pracy. Jednakże równie niekorzystny jest niedostatek nadrzędnych koncepcji i powiązania prac nad specjalistycznym oprogramowaniem, prowadzonych w rozmaitych ośrodkach przy znacznym rozproszeniu sił i środków.

Niniejszy artykuł powstał w związku z próbami koordynacji prac nad automatyzacją projektowania w przemyśle maszynowym. Wydaje się jednak, że wiele wyciągniętych stąd wniosków stosuje się także do działalności projektowej w innych gałęziach przemysłu.

Tradycyjne formy oprogramowania dla prac projektowych

Podstawową przyczyną skłaniającą projektantów i konstruktorów do posługiwania się komputerami jest ich duża skuteczność w rozwiązywaniu cząstkowych problemów projektowania, np. posługiwaniu się skomplikowanymi modelami matematycznymi, obliczeniach wariantowych i optymalizacyjnych, wyszukiwaniu informacji. Programy stosowane w projektowaniu inżynierskim można podzielić na trzy klasy:

1. programy dla zagadnień specjalnych
2. programy zorientowane na określone obiekty techniczne
3. programy zorientowane na ogólnie sformułowane problemy.

Programy pierwszej grupy są opracowywane dla powstających sporadycznie zadań, których rozwiązanie wymaga dużego nakładu obliczeń. Okres użytkowania jest krótki, liczba użytkowników jest z reguły niewielka.

Programy lub pakiety programów zorientowane obiektowo opracowuje się dla prac projektowych dotyczących konkretnego obiektu technicznego. Automatyzują one powtarzalne czynności związane z zagadnieniami przewidywanego wytwarzania i użytkowania obiektu, obliczenia inżynierskie, analizę kosztów, wyszukiwanie i przetwarzanie informacji.

Programy obiektowe powstają zwykle w placówkach związanych organizacyjnie z producentem obiektu. Pracochłonność ich przygotowania jest często duża, zdarzają się opracowania o pracochłonności ponad 10 roboczo-lat. Również koszt konserwacji programów obiektowych jest wysoki i sięga przy pięcioletnim okresie użytkowania 50—100% początkowego kosztu.

Korzyści ekonomicznych z ich stosowania można oczekiwać tylko wtedy, gdy zastosowania mogą być starannie planowane i realizowane i gdy częstotliwość użytkowania jest duża. Jest to praktycznie osiągalne jedynie w placówkach projektowych specjalizujących się w wąskiej tematyce technicznej, w których nie ma barier psychologicznych utrudniających stosowanie techniki obliczeniowej.

Programy lub pakiety programów zorientowane problemowo realizują zwykle pewną metodę rozwiązania zagadnienia powtarzającego się przy różnych obiektach technicznych. Charakterystycznymi problemami podejmowanymi w tej formie są:

— symulacja procesów ciągłych lub dyskretnych
— obliczenia wytrzymałościowe
— obliczenia własności materiałowych
— obliczenia optymalizacyjne
— przygotowanie rysunków technicznych
— przygotowanie nośników informacji dla urządzeń produkcyjnych sterowanych numerycznie.

Stosowanie programów tej grupy do określonych zagadnień projektowania wymaga przede wszystkim sformułowania problemu według szablonu założonego w programie, co bywa nieraz trudne i bardzo pracochłonne. Mniej kłopotliwe jest już później dostarczenie odpowiednio przygotowanych danych liczbowych, zaprogramowanie elementów modelu matematycznego itp. Dlatego też programy zorientowane problemowo są stosowane właściwie tylko tam, gdzie działa zespół ludzi specjalizujących się w samym problemie. Dobrze opracowane programy problemowe stanowią rzadkość. Mogą one powstać tylko w wyspecjalizowanych placówkach zajmujących się oprogramowaniem, jako wytwór świadomie zaplanowanej i zorganizowanej działalności, uwolnionej od presji bezpośrednich efektów praktycznych. Tworzenie takich programów „przy okazji" projektowania określonych obiektów jest wręcz niemożliwe. Nakład pracy potrzebny na postawienie problemu w ogólnej formie i opracowanie uniwersalnej metody rozwiązania jest bardzo duży i może sięgać kilkunastu roboczo-lat.

Systemy oprogramowania do prac projektowych

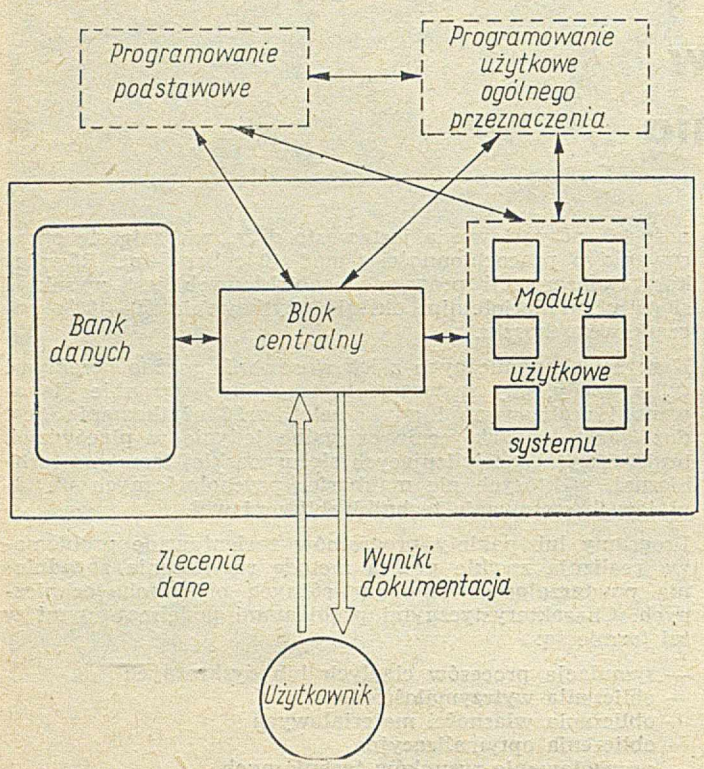
Rozpowszechnienie klasycznych form oprogramowania dla prac projektowych jest utrudnione ich wysoką pracochłonnością. Szczególne trudności powstają w placówkach projektowych i konstrukcyjnych o zmiennym profilu rzeczowym i w placówkach nie dysponujących własną, wyspecjalizowaną kadrą informatyczną. Doświadczenia wysoko rozwiniętych krajów uprzemysłowionych wykazują jednak, że trudności te można pokonać koncentrując prace na zintegrowanych systemach oprogramowania, które mogą służyć dużej liczbie użytkowników [1, 2, 3]. Systemy takie zawierają zazwyczaj:

— blok centralny
— moduły zorientowane problemowo
— moduły zorientowane obiektowo.

Moduły zorientowane obiektowo spełniają w ramach systemu rolę analogiczną do roli niezależnych programów obiektowych. Jednakże przy odpowiedniej strukturze systemu, autorzy modułów obiektowych mają do dyspozycji uniwersalne moduły problemowe, które mogą być przyłączane do różnych jednostek funkcjonalnych. Natomiast blok

Dr inż. KRZYSZTOF URBANIEC ukończył Wydział Mechaniczny Energetyki i Lotnictwa Politechniki Warszawskiej (1964), gdzie również uzyskał doktorat (1970). Kieruje Pracownią Modelowania Matematycznego Instytutu Techniki Cieplnej PW i współpracuje z Ośrodkiem Wiodącym Automatyzacji Prac Inżynierskich przy CHEMADEX — Warszawa. Autor 30 publikacji, a także szeregu programów inżynierskich wykorzystywanych w przemyśle.

centralny przejmuje i w ujednolicony sposób wykonuje takie czynności, jak:
— wprowadzanie danych i edycja wyników
— gospodarowanie zbiorami informacji



Rys. Schemat ideowy oprogramowania systemu projektowania

— reakcje na błędy
— dokumentacja prac
— sterowanie tokiem dłuższych ciągów czynności.

Efektywna realizacja czynności bloku centralnego wymaga na ogół oprogramowania w językach typu assembler, związanego z typem i częściowo konfiguracją sprzętu. Natomiast moduły użytkowe mogą być programowane w językach wyższego rzędu, co pozwala uzyskać znaczną ich niezależność od sprzętu.

Moduły zorientowane obiektowo zwykle mogą być użytkowane w pojedyńczych przedsiębiorstwach lub grupach przedsiębiorstw, a moduły zorientowane problemowo — w jednej lub kilku branżach przemysłowych. Bloki centralne systemów projektowania nie są ściśle związane z zainteresowaniami użytkowników i mogą być używane nawet w zupełnie różnych branżach. Dzięki temu możliwa jest koncentracja prac nad najtrudniejszymi elementami oprogramowania, a poszczególni użytkownicy mogą tworzyć indywidualne systemy, dołączając do jednakowych bloków centralnych różne zestawy modułów użytkowych. Rozwiązania przyjęte dla bloku centralnego powinny zresztą być podporządkowane zasadzie systemu otwartego, do którego użytkownicy mogą dołączyć własne moduły użytkowe.

Warunki realizacji systemów

Organizacja modularna jest dziś powszechnie stosowana dla oprogramowania podstawowego i użytkowego nowoczesnych komputerów. Dzięki temu, w pracach nad systemami dla projektowania można zaadaptować i wykorzystać wiele istniejących rozwiązań, typowych programów organizacyjnych, itp. Muszą to być jednak rozwiązania dające możliwości bardzo elastycznego wykorzystywania modułów, co stwarza wysokie wymagania wobec stosowanego sprzętu i jego oprogramowania firmowego. W Polsce dysponujemy już takim, odpowiednio oprogramowanym sprzętem: wystarczy wspomnieć komputery serii ODRA pod kontrolą systemu operacyjnego GEORGE, IBM 360/370 pod kontrolą OS lub DOS, czy minikomputery PDP 11/45 z systemami RSX-11D lub IAS. Również parametry techniczne i koncepcje oprogramowania maszyn Jednolitego Systemu stwarzają możliwości efektywnej realizacji systemów projektowania.

Przy konfiguracji sprzętu i zestawu oprogramowania firmowego mogą być różne w różnych zastosowaniach [5, 6]. Przykładowo, w dużych systemach integrujących projektowanie i zarządzanie występuje problem gospodarowania bardzo dużymi zbiorami informacji.

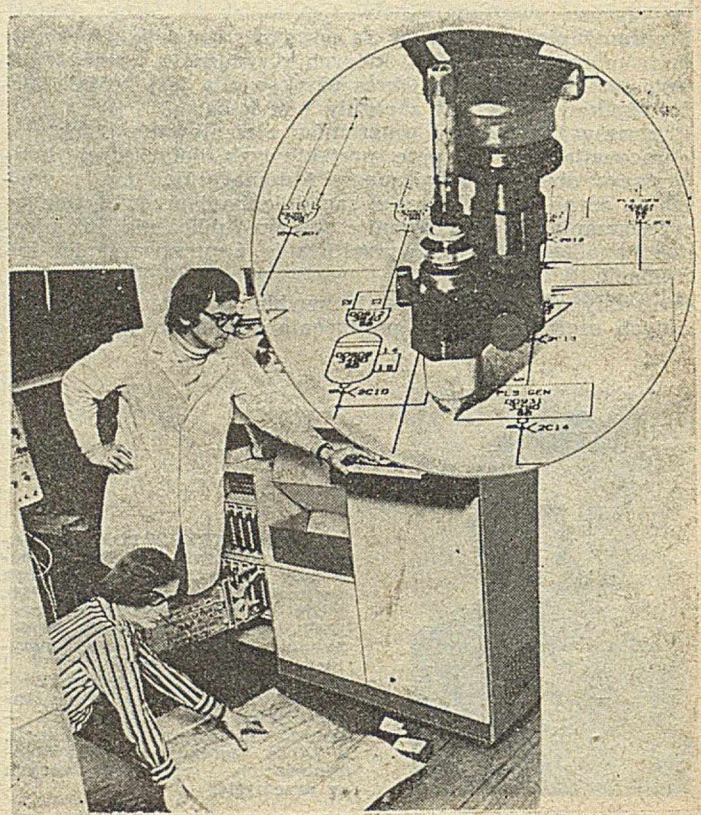
Ponieważ pojedyńcze, nierozdzielne bloki mogą dochodzić do miliona bajtów [7], efektywne wykonanie wielu typowych czynności wymaga dużej, odpowiednio zorganizowanej pamięci operacyjnej. W projektowaniu często stosuje się graficzny zapis informacji, a więc ważna rola może przypadać urządzeniom graficznym [7, 8]. W wielu zastosowaniach bardzo ważny jest konwersacyjny tryb wykorzystania programów [7, 9]. Umożliwia on projektantom pełną koncentrację na rozwiązywanym problemie, bez przerw na przygotowanie tradycyjnych nośników informacji.

Nie rozwijając nadmiernie tego tematu, warto przytoczyć przykłady. W obliczeniach wytrzymałości maszyn (również budowli, statków i innych obiektów) szerokie zastosowanie znajduje tzw. metoda elementów skończonych. Wyniki obliczeń wykonanych tą metodą są najbardziej czytelne wtedy, gdy mają postać graficzną. Automatyzacja kreślenia skraca czas opracowania dużych problemów o ponad 30%, przy jeszcze wyraźniejszej redukcji kosztów. Metoda elementów skończonych charakteryzuje się również dużą pracochłonnością przygotowania danych, w których łatwo popełnić omyłki. Zastosowanie interakcyjnych urządzeń graficznych do kontroli danych i wprowadzania poprawek pozwala skrócić czas opracowania dużych problemów o 50%, skutecznie eliminując przy tym możliwość „przepuszczenia" błędów w wynikach [10].

Przegląd doświadczeń

Pionierskim osiągnięciem w dziedzinie zintegrowanych systemów oprogramowania dla prac projektowych był system ICES, opracowany w USA głównie dla potrzeb projektowania konstrukcji budowlanych [11]. Koncepcja tego systemu powstała w 1964 r. i została pozytywnie oceniona przez licznych użytkowników. Niektóre podsystemy systemu ICES są dostępne w Polsce w ośrodkach wyposażonych w komputery IBM 360/370.

Zasadnicze założenia systemu ICES wykorzystano w pracach nad systemami: MODCON w USA [12], GENESYS w Anglii [13], ISB i REGENT w RFN [14, 15]. Wiele zbliżo-



nych rozwiązań można też znaleźć w norwerskich systemach BOSS i PRELIKON [1] dla projektowania statków. Dotychczasowe prace zaowocowały w postaci wielu cennych koncepcji teoretycznych lub nawet uniwersalnych podsystemów, które mogą być przydatne w nowych opracowaniach. Dotyczy to np. gospodarowania dużymi bankami danych [16], przekazywania informacji między modułami języków problemowych [18, 19], czy koordynacji działania wielu modułów w oparciu o tablice decyzyjne [20, 21].

Ważne są praktyczne doświadczenia z wykorzystania oprogramowania w różnych systemach projektujących (rozumianych jako zbiór ludzi, sprzętu i metod). W pracach nad oprogramowaniem trzeba np. uwzględniać powiązania systemu projektującego z jego otoczeniem, przejawiające się w istnieniu przepisów bezpieczeństwa, norm materiałowych, itp. Gdy system służy projektowaniu obiektów podlegających instytucjom dozoru technicznego (zbiorniki ciśnieniowe, budynki itp.) metody obliczeniowe muszą być zgodne z aktualnymi przepisami. Staje się więc celowe, aby instytucje dozoru kontrolowały niektóre moduły systemów projektowania.

Z wielu zastosowań wynikły postulaty zmiany przepisów, utrudniających wprowadzenie komputerowych technik projektowania. Np. tradycyjne normy rysunku technicznego zawierają zasady pisma technicznego i różnicowania grubości linii, które trudno zrealizować na plotterach. Ponadto, normy materiałowe, układy pasowań i inne ważne informacje niezbędne w projektowaniu są publikowane w niewygodnej formie tabelarycznej lub graficznej. Wydaje się, że można opracowywać je od razu także w postaci zależności aproksymacyjnych lub odpowiednich algorytmów wyszukiwania.

Interesujące jest spostrzeżenie, że istnieją naturalne granice wielkości systemów dla prac projektowych. W dużym systemie zbyt trudna może być ciągła aktualizacja zbiorów danych i algorytmów w miarę postępu technicznego, zmian przepisów i ustawodawstwa, zmian sytuacji ekonomicznej itp. Koszt opracowania i użytkowania systemów rośnie mniej więcej wykładniczo z liczbą modułów [1]. Ponadto, zapobieżenie propagacji błędów przy współpracy wielu modułów jest bardzo kłopotliwe. Pocieszające jest jednak to, że zastosowania potwierdziły również użyteczność skromnych rozmiarami systemów o bardzo prostej organizacji [22, 23].

Ponieważ polskie prace nad uniwersalnymi komputerowymi systemami projetowania nie wyszły dotąd poza fazę wstępną, warto zwrócić uwagę na to, jak organizowano takie prace w innych krajach.

Wszędzie stosowano regułę uniezależnienia prac od bezpośrednich efektów praktycznych. System ICES został opracowany w Masschussets Institute of Technology w ramach prac badawczych finansowanych dotacjami agend rządowych. Specjalne przedsięwzięcia badawcze organizowano w RFN dla systemów ISB i REGENT, w Norwegii dla systemu BOSS, w Anglii dla prac prowadzonych w CAD-Laboratory w Cambridge. Nie przeszkodziło to, że systemy znalazły potem bardzo licznych użytkowników. Wypróbowane w nich koncepcje stały się podstawą dalszych prac badawczych i wielu wdrożeń. Rozwój systemów automatyzacji projektowania odbywa się dziś w tych samych ośrodkach, w których powstają nowe kierunki rozwoju światowej techniki [24, 25, 26, 27].

LITERATURA

[1] Vahl T.: Present and future possibilities within computeraided design. Computers and Structures, vol. 4, nr 1, 1974 r.

[2] Schlechtendahl E. G.: Problemem der EDV — Anwendung in der Industrie. Konstruktion, vol. 25, nr 4, 1973.

[3] Mirowski W.: Automatyzacja obliczeń zawodowych — inżynierskich. Mat. Telewizyjnego Kursu Informatyki, OBRI, Warszawa 1973.

[4] Wolverton R. W.: The cost of developing large-scale software. IEEE Trans. on Computers, vol. C-23, nr 6, 1974.

[5] Bąbiński Cz.: Systemy projektujące. W zbiorze „Metodologia projektowania inżynierskiego" pod red. W. Gasparskiego, PWN, Warszawa 1973.

[6] Brankamp K.: Wie beginnt man mit der Rationalisierung im Kontruktionsbereich? VDI-Berichte, nr 191, 1973.

[7] Parslow R. P., Green R. E. (ed): Advanced Computer Graphics. Planum Press, London (New York 1971).

[8] Baatz W.: Erstellung von Einzelteilzeichnungen mittels graphischer DVA. Konstruktion, vol. 24, nr 9, 1972.

?9] Schreiner R. N., Frild B. D.: An interactive graphical computer system as a portal for perceiving engineering problems. Computers and Structures, vol. 1, nr 2, 1971.

[10] Bousquet R. D., Yates D. N.: A low cost interactive graphics system for large scale finite element analyses. Computers and Structures, vol. 3, s. 1321—1330, 1973.

[11] Roos D.: ICES — system, general description. MIT, Dept. of Civil. Engn., R 67—49, 1967.

[12] Rien R. E.: The development of a preliminary concept of the MODCON Operating System. CAD — Lab. Dept. of Architectural Engn. Pensylvania State Univ. R 70—5, 1970.

[13] Alcock J., et al.: GENESYS Reference Manual. University of Loughbotough, 1971.

[14] Pahl P. J.: Der Aufbau des Informationssystems für das Bauwesen. Deutscher Betontag, Berlin 1971.

[15] Enderle G., et al.: Erster REGENT — Haljbjahresbericht. KFK — Nachrichten vol. 4, nr 8, 1972.

[16] Senko M. E.: Data structures and accessing in data-base systems. IMB Systems Journal, vol. 12, nr 1, 1973.

[17] Hatfield F. J.: The Information Organizer — a system for symbolic data manipulation. Computers and Structures, vol. 1, nr 1, 1971.

[18] Lopez L. A.: POLO — problem oriented language organizer. Computers and Structures, vol. 1, s. 555—572, 1972.

[19] Problem Language Organizer. Program IBM 360 A-CX-26X.

[20] Fenves S. J.: Representation of the computer — aided design process by a network of decision tables. Computers and Structures, vol. 3, s. 1099—1107, 1973.

[21] Burgielski B.: Wykorzystanie tablic decyzyjnych w programowaniu oraz rozwiązywaniu problemów typu projektowego. Informatyka, vol. 10, nr 6, 1974.

[22] Mischke C. R.: An introduction to computer-aided design. Prentice Hall, Englewood Cliffs 1968.

[23] Whitney D. E., Milley M. K.: CADSYS — a new approach to computer aided design. IEEE. Trans. on Systems, Man and Cybernetics, vo. SMC — 4, nr 1, 1974.

[24] Engineering Design Data Processing at IBM. IBM Data Processing Application, GE 20-8155-0, 1970.

[25] English C. H.: Computer aided design-drafting (CADD) — engineering (manufacturing tool. J. of Aircraft, vol. 10, nr 12, 1973 r.

[26] Crookes W. A.: et al.: CAPS — a computer based aid to aircraft project studies. Computer Aided Design, vol. 6, nr 2, 1974.

[27] Fulton R. E., McComb H. G.: Automated Design of Aerospace Structures. Trans. ASME, J. of Engineering for Industry, vol. 95, February 1974.

Autor dysponuje bibliografią przedmiotu, złożoną z ponad 100 pozycji.

WACŁAW ISZKOWSKI
Instytut Maszyn Matematycznych
Politechnika Warszawska

681.322.004.14:65.012.432

Organizacja i wyszukiwanie informacji za pomocą metod transformacyjnych

Przegląd metod transformacji służących organizacji i wyszukiwaniu informacji w stosowanych metodach programowania i przetwarzania zbiorów. Opisano podstawy metody transformacji, transformację pierwotną i wtórną, zgrupowanie wtórne, przekształcanie zakodowanych tablic HASH oraz sposoby określania efektywności tych metod.

W aktualnie stosowanych metodach programowania i przetwarzania zbiorów, a także w pracy translatorów i systemów operacyjnych około 40 procent czasu działania programu jest poświęcone na zapisanie, lub wyszukanie informacji. Klasyczne metody przeszukiwania tablic charakteryzują się jednak bardzo małą efektywnością.
Najbardziej znaną i nojpowszechniej używaną jest metoda PRZESZUKIWANIA LINIOWEGO polegająca na kolejnym czasochłonnym przeglądaniu elementów tablicy, aż do znalezienia żądanej informacji. Następną metodą jest PRZESZUKIWANIE BINARNE polegające na uprzednim posortowaniu tablicy według wzrastającej wartości elementu charakterystycznego zbioru. Wyszukiwanie danej pozycji sprowadza się do dzielenia tablicy na coraz mniejsze obszary, które wybierane są kolejno w wyniku działania operatora relacji, aż do uzyskania żądanej pozycji, co następuje przeciętnie po $log_2 N$ krokach (N-rozmiar tablicy). Metoda ta jest niewygodna przy kolejno wykonywanym wyszukiwaniu i wprowadzaniu nowych pozycji, gdyż wymaga każdorazowo powtórnego przesortowania tablicy.

Podstawy metod transformacji

Zbiór informacji umieszczony jest w tablicy podzielonej na poszczególne pozycje zawierające elementy zbioru, z których każdy jest określony jednoznacznym charakterystycznym dla niego identyfikatorem. Wprowadzenie nowych elementów do zbioru, a więc jego tworzenie polega na umieszczeniu w tablicy identyfikatora i informacji dotyczących elementu, czyli tak zwanej treści informacyjnej. Z kolei otrzymanie informacji o elemencie sprowadza się do wyszukiwania w tablicy podanego dla niego identyfikatora. Oba rodzaje operacji na zbiorze mogą być wykonywane wzajemnie naprzemian. Identyfikatory występują w postaci alfanumerycznej prostej, lub złożonej z kilku części opisujących dany element bardziej lub mniej szczegółowo w kontekście całego zbioru. Ilości kombinacji wszystkich możliwych symboli przy danej postaci identyfikatora jest znacznie większa od liczności zbioru informacji. Przykładem identyfikatorów mogą być:
— numery kont bankowych, np. XXXX—XXXXXX, którego pierwszy czterocyfrowy człon oznacza numer oddziału, a następny numer konta klienta w oddziale
— kody materiałowe, np. Systematyczny Wykaz Wyrobów SWW
— numery statystyczne przedsiębiorstw
— numery katalogowe księgozbiorów
— numery identyfikujące pracowników w przedsiębiorstwie (itd.),
a dla oprogramowania maszyn cyfrowych:
— kody literowe rozkazów języka programowania w translatorach
— nazwy zbiorów organizowanych przez System Operacyjny na dyskach lub bębnach
— współrzędne tablic, w których nie są wykorzystane wszystkie elementy (tzw. tablice rozproszone itd.).
Przedstawione wyżej przykłady obrazują jednocześnie możliwości zastosowań metod transformacji.

Wykorzystując identyfikatory jako podstawę metody wyszukiwania zamienia się je na klucze, które w maszynie występują jako liczby całkowite. Najczęściej jest to odwzorowanie tożsamościowe, chociaż przy postaci alfanumerycznej identyfikatora należy go najpierw zdekodować w postać numeryczną i ewentualnie wykorzystać tylko niektóre jego pozycje przy tworzeniu klucza. Wspomniane we wstępie metody wyszukiwania informacji dla tych zastosowań wykazują bardzo niską efektywność, gdyż wymagają długotrwałego przeszukiwania tablicy, zwiększającego się ze wzrostem liczby elementów zbioru informacji.
Metoda transformacji pierwotnej polega na bezpośrednim jednoznacznym przekształceniu wartości klucza w adres względem początku tablicy przy wprowadzaniu nowej pozycji do zbioru. Analogicznie, wyszukanie żądanej, uprzednio wprowadzonej informacji identyfikowanej znanym kluczem (lub identyfikatorem), sprowadza się do wykonania identycznego przekształcenia. Omówione niżej transformacje mogą czasami prowadzić do określenia tego samego adresu dla różnych kluczy, co powoduje tzw. „zderzenia". Wykonanie, czasami kilkakrotne, transformacji wtórnej polegające na wyznaczeniu nowego adresu z adresu dającego „zderzenie", pozwala na znalezienie adresu wolnej pozycji na zapisanie informacji, lub na szybkie określenie jej pozycji. Zaletą metody transformacji jest głównie znaczna szybkość wyszukiwania pozycji i to bez względu na wielkość zbioru. Podstawową zasadą metody jest stosowanie jednakowych transformacji pierwotnych i wtórnych dla zapisu i odczytu dla wszystkich elementów tworzonego zbioru. Po wprowadzeniu przynajmniej jednego elementu do zbioru, wszelkie zmiany jego struktury (zmiana rozmiaru, przesunięcie pozycji), mogą być zrealizowane tylko przy pomocy specjalnych algorytmów i wiążą się z przeliczeniem adresów wszystkich pozycji według nowych parametrów. Przedstawiona na rysunku 1 sieć działań obrazuje algorytm stosowania metody transformacji.

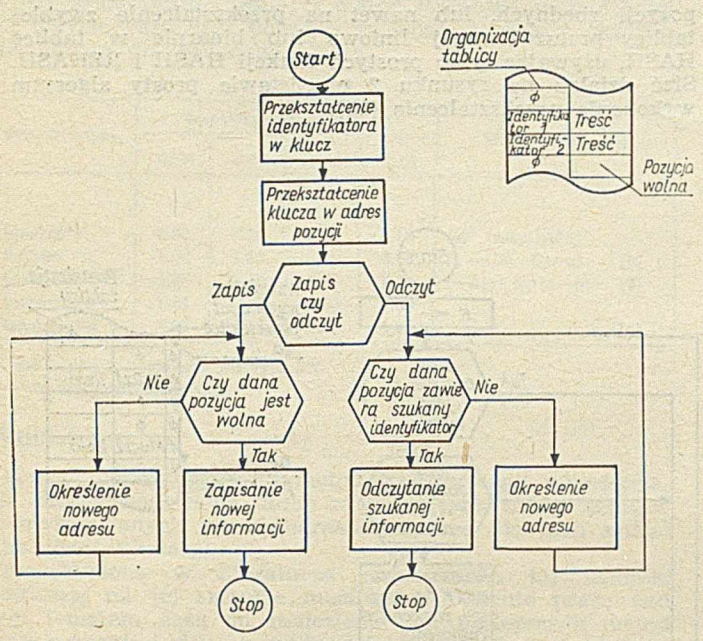
Transformacja pierwotna

Wprowadźmy dla transformacji pierwotnej definicje funkcji HASH [1] (lub SCRAMBLE) przekształcającej klucz w adres, określoną następująco:

1) HASH — amer. mielonka, siekać, przen. bigos, bałagan, galimatias
SCRAMBLE — ang. jajecznica
REHASH — amer. przesiekać; Ze względu na brak odpowiednika w termiologii polskiej w dalszej części artykułu będziemy używać nazwy oryginalnej HASH i REHASH.

Mgr inż. WACŁAW ISZKOWSKI ukończył Wydział Elektroniki Politechniki Warszawskiej, kierunek maszyny matematyczne (1972 r.). Obecnie pracuje w Instytucie Maszyn Matematycznych Politechniki Warszawskiej jako starszy asystent i zajmuje się opracowaniem i realizacją systemów operacyjnych.

Uwzględnienie rozmiaru tablicy przy obliczaniu funkcji jest związane z technicznymi i praktycznymi ograniczeniami rozmiaru tablicy. Podstawową więc metodą transformacji jest zmniejszenie wartości klucza do takiej wartości, aby mieściła się ona w zakresie tablicy. Realizacja metody zależy od sposobu działania na liczbę reprezentującą klucz, czyli od używanego języka programowania.



adres: = HASH (klucz [mod (rozmiar tablicy)]

Metodą klasyczną jest podzielenie wartości klucza przez rozmiar tablicy i pobranie reszty jako adresu względem jej początku. Ze względu na własności dzielników i reszt, dla otrzymania wszystkich możliwych adresów z danego zakresu wykazano, że podzielnikiem, a więc rozmiarem tablicy powinna być maksymalna liczba pierwsza mniejsza od 2^N, aby jednocześnie wykorzystać prawie wszystkie kombinacje N-bitów. Na przykład realizacja w ALGOLu-60 jako procedury funkcyjnej **integer procedure** HASH (klucz, rozmiar); przedstawia się następująco:

HASH: = RESZTA (klucz, rozmiar) + 1;

gdzie **integer procedure** RESZTA (A, B); realizuje się jako:

RESZTA: = A — entier $(A/B) \times B$;

Przy wykorzystaniu języków wewnętrznych programowania, funkcje realizuje się przez prostą operację wycięcia ze słowa binarnego reprezentującego dany klucz N bitów i używa się wobec tego tablicy o rozmiarze 2^N [5]. Ewentualnie identyczny efekt otrzyma się sztucznie, wykorzystując języki programowania wyższego rzędu:

HASH: = RESZTA (klucz, 2↑N) + 1;

Analizując poszczególne zbiory identyfikatorów, stwierdza się, że często niektóre pozycje są identyczne dla całego zbioru, lub charakteryzują go bardzo ogólnie. Występuje to zwłaszcza w przypadku realizacji zbioru informacji będącego podzbiorem większego systemu informacyjnego.

Na przykład: dla zbioru kont bankowych w pojedynczym oddziale, wystarcza jako wartość klucza pobrać z identyfikatora tylko ostatnie sześć cyfr oznaczających numery kont klientów.

W wielu też przypadkach dla identyfikatorów bardziej złożonych, dla wzmocnienia wpływu poszczególnych ich pozycji, wykonuje się wstępne operacje przetwarzania wartości identyfikatora w klucz, na przykład przez [6]:

— wycinanie środkowych cyfr, lub bitów z kwadratu liczby reprezentującej identyfikator:

klucz: = identyfikator↑2)/2↑(M/4));

gdzie M określa liczbę bitów słowa maszynowego

— zsumowanie części górnej słowa binarnego reprezentującego identyfikator z jego częścią dolną:

klucz: = RESZTA (identyfikator, 2↑ (M/2)) + entier (identyfikator) (2↑(M/2));

— kontatenację niektórych jego pozycji,

— kolejne konwersje o różnych podstawach: dwójkowych, ósemkowych lub dziesiętnych (itp).

Wybór jednej z metod zależy od cech charakterystycznych identyfikatorów, posiadanej maszyny i od używanego języka programowania.

Przykładowo, dla zbioru kont jednego oddziału, w którym mamy mniej niż 509 (lub 512 = 2^9) klientów i dla numeru konta 0121 — — 254371 bierze się jako klucz wartość 254371, którą transformuje się w adres równy: 381=RESZTA (254371, 509) + 1.

Transformacja wtórna

Po wykonaniu funkcji HASH sprawdza się pozycję o obliczonym adresie. W przypadku jej zajętości, lub jeśli istniejący w niej idetyfikator jest inny od poszukiwanego, powstaje zderzenie (nadmiar, synonim, kolizja) i adres jest pierwotnym adresem kolizyjnym. Określenie nowej pozycji polega na dodaniu inkrementu jako przyrostu do tego adresu i na analogicznym jego zbadaniu. Proces ten w przypadku negatywnego wyniku może być powtarzany wielokrotnie, aż nawet do przeglądnięcia całej tablicy (występuje wtedy przepełnienie zbioru). Kolejne adresy dające zderzenia nazywa się adresami kolizyjnymi. Ogólnie transformację wtórną jako funkcję można przedstawić w ALGOL-u 60 w postaci **integer procedure** REHASH (adres, rozmiar);, gdzie adres jest adresem kolizyjnym lub pierwotnie kolizyjnym w zależności od przyjętego algorytmu.

adres: = REHASH (adres kolizyjny, rozmiar) [mod rozmiar tablicy]:

Realizację funkcji czyli dodanie przyrostu wykonuje się następująco:

adres: = pierwotny adres kolizyjny + przyrost:

lub

adres: = adres kolizyjny + przyrost;

Przykładowo, dla numerów kont dla klucza 254371 i 151553 otrzyma się ten sam adres 381, który dla drugiego z kluczy będzie pierwotnym adresem kolizyjnym, gdyż pozycja 381 została już zajęta przez informację identyfikowaną kluczem 254371.

Metody realizacji tej funkcji są różnorodne i one w dużym stopniu ze względu na swoją czasochłonność wpływają na efektywność danej transformacji. Wybór jednej z nich jest podyktowany tymi samymi warunkami, jak dla funkcji HASH.

Najprostszą realizacją jest metoda liniowa polegająca na dodaniu do adresu jako przyrostu ustalonej wartości stałej CONST.(6). Rozmiar tablicy równa się liczbie pierwszej, a poszukiwanie wolnej lub żądanej pozycji kończy się po jej znalezieniu, lub po przeglądnięciu całej tablicy:

REHASH: = RESZTA (adres kolizyjny + CONST, rozmiar) + 1;

Metoda ta powoduje jednak powstawanie zgrupowań pierwotnych, określonych jako nakładanie się części elementów ciągów adresów powstałych w wyniku kolejnego działania funkcji REHASH dla różnych pierwotnych adresów kolizyjnych.

Przykładowo, dla podzbioru kluczy o adresie pierwotnym 381 i podzbioru o adresie 386 dla CONST=5 otrzyma się zgrupowanie pierwotne dla ciągów adresów 381, 386, 391, 396 ... i 386, 391, 396, 401 ...

Następne metody likwidują zgrupowania pierwotne przez niedopuszczania do powstania identycznych w pewnym zakresie ciągów adresów. Używając repetycyjnego generatora pełno-okresowego liczb pseudo-losowych, otrzyma się kolejne przyrosty. Ze względu na zasadę działania generatora, rozmiar tablicy musi być równy 2^N, a początkowa liczba generująca LGEN powinna mieć wartość równą 1, tak aby przy każdym nowo rozpatrywanym kluczu, generator wytwarzał ten sam ciąg przyrostów. Realizacja metody jest następująca [6, 19]:

LGEN: = RESZTA (5 × LGEN, 2↑(N+2));

REHASH: = RESZTA (pierwotny adres kolizyjny + entier (LGEN/4, 2↑N) + 1;

Przy użyciu języka wewnętrznego rozkazowe bezpośrednie wycinanie ze słowa odpowiedniej liczby bitów znacznie zwiększa szybkość metody.

Przykładowo, dla N równego 9 (rozmiar tablicy równy 512) i dla pierwotnego adresu kolizyjnego 491 otrzymujemy następujący ciąg adresów i generatora 493, 498, 11, 8, 507...

Kolejna metoda, kwadratowa, polega na dodawaniu przyrostu równego: INCR = CONSTA × I + CONSTB × I ↑2, gdzie I przyjmuje kolejno wartości 1, 2, 3, ..., (rozmiar 2). Dokładniejsza analiza funkcji prowadzi do jej uproszczenia INCR = I↑2 bez wpływu na końcową efektywność. Dla tej metody [19] rozmiar tablicy jest liczbą pierwszą i przeszukiwane jest tylko (rozmiar—1)/2 pozycji, jakkolwiek przy użyciu niewielkiej modyfikacji możemy dany przyrost

wykorzystać dwukrotnie do określenia kolejno dwóch różnych pozycji leżących w uzupełniających się przeszukiwanych zbiorach. Zamieszczony algorytm pozwala na wyszukiwanie naprzemian adresów z obu zbiorów, przy czym przed pierwszym wywołaniem funkcji $I = 1$, a zmienna logiczna ZMLOG: = **true**;:

```
pom: = RESZTA (pierwotny adres kolizyjny + I↑2,
rozmiar);
if ZMLOG then ZMLOG: = false
else begin
pom: = RESZTA (2 × pierwotny adres kolizyjny +
+ rozmiar — pom, rozmiar);
ZMLOG: = true;
I; = I + 1 end;
REHASH: = pom + 1;
```

Z kolei dla zwiększenia efektywności metody pod względem szybkości obliczania adresu, można dodawać jako przyrost jedynie różnice między dwoma kolejnymi inkrementami metody kwadratowej obliczonymi dla I i $I + 1$. Pozwala to uniknąć dzielenia przez rozmiar tablicy, gdyż obliczony adres jest zawsze mniejszy od podwójnej wartości rozmiaru tablicy, a więc może być zmniejszony przez odejmowanie.
Początkowo INCR przyjmuje wartość (—rozmiar):

```
INCR: = INCR + 2;
pom: = adres kolizyjny + abs (INCR);
REHASH: if pom > rozmiar then pom - rozmiar
else pom;
```

Przedstawione algorytmy spełniają szereg czynników koniecznych do zapewnienia skutecznego i szybkiego obliczania adresu pozycji dla danego klucza.
Nieco odmienną jest metoda łańcuchowa [6, 19], której zasada opiera się na utworzeniu dodatkowej tablicy nadmiaru, do której wprowadzane są pozycje, dla których wystąpiło zderzenie. Jednocześnie przy każdej pozycji umieszcza się wskaźnik do następnej w danym łańcuchu zawierającym kolejne pozycje jednego z podzbiorów kluczy. Metoda ta daje możliwość zwiększenia pola dla nowych kluczy, bez konieczności przeliczania adresów kluczy już wprowadzonych. Często tablica nadmiaru co do rozmiaru równa jest tablicy podstawowej, w tym jednak przypadku czas wyszukania danej pozycji znacznie się wydłuża.

Zgrupowania wtórne

W zadanych zbiorach kluczy występują często podzbiory określone tym samym pierwotnym adresem kolizyjnym. Ostatni z rozpatrywanych elementów takiego podzbioru wymaga dla określenia jego pozycji, wykonania tyle razy funkcji REHASH, ile jest wszystkich w nim elementów. Takie zgrupowania wtórne adresów powodują wydłużenie czasu wyszukiwania. Metodą otrzymania różnych ciągów dla tych samych pierwotnych adresów kolizyjnych, jest takie zmodyfikowanie funkcji, aby uzależnić przyrost od wartości klucza. Najprostszym działaniem jest wykorzystanie obliczonego w funkcji HASH ilorazu wartości klucza przez rozmiar tablicy — ILORAZ, jako przyrostu dla funkcji REHASH [12, 18]. Ta prosta metoda znacznie zmniejsza czas obliczeń, a jednocześnie likwiduje zgrupowania wtórne:

```
REHASH: = RESZTA (adres kolizyjny + ILORAZ,
rozmiar) + 1;
```

Również metoda kwadratowa [8] może być zmodyfikowana przez zastąpienie stałej CONSTB ILORAZ-em. Zakładając uproszczoną formę funkcji kwadratowej, funkcje REHASH realizuje się nadając zmiennej INCR wartość równą (—rozmiar) przed jej pierwszym wywołaniem, po obliczeniu wartości funkcji HASH:

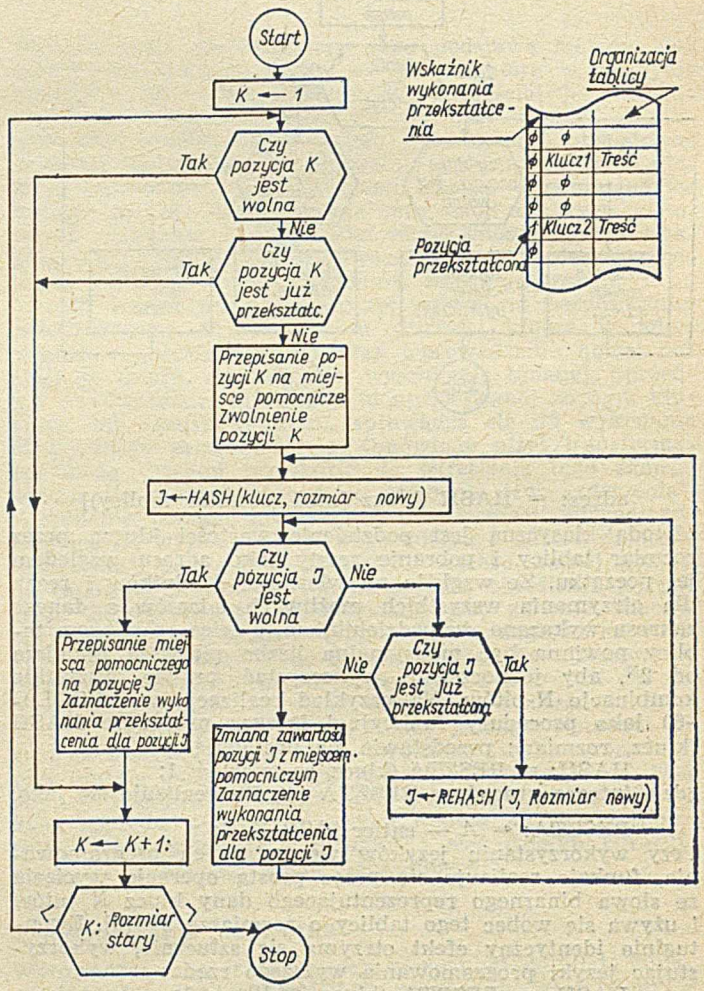
```
INCR: = INCR + 2;
REHASH: = RESZTA (abs(pierwotny adres kolizyjny + INCR × ILORAZ), rozmiar) + 1;
```

Metoda ta jednocześnie eliminuje zgrupowania pierwotne i wtórne.

Przykładowo, dla kluczy 254311, 151553, 156643, 471205 (rozmiar 509) otrzyma się ten sam pierwotny adres kolizyjny 381. Stosując metodę ilorazowo-kwadratową, dostanie się w wyniku pojedynczej transformacji wtórnej następujące adresy pozycji: 254311 = = 381, jako pierwszy klucz z podzbioru) 151553 = RESZTA (abs (381—507×297), 509)+1 = 44, 156643 = 25, 471205 = 315.

Przekształcenie zakodowanych tablic HASH

Utworzone tablice ze względu na swą specyficzną formę powstawania nie mogą być zmieniane co do rozmiaru, lub struktury. Wymazanie jakiejkolwiek pozycji powodowałoby zmiany w ciągach adresów kolizyjnych, a tym samym mogłoby uniemożliwić odszukanie pozycji z danego ciągu. Stosowanie dodatkowych znaczników powoduje oczywiście wzrost czasu wyszukiwania, pozostaje więc tylko, jak i dla zmiany rozmiaru tablic, konieczność przeliczenia powtórnie, wszystkich zapisanych w niej pozycji. Wykonanie relokacji w przypadku zmian pochłania pewien okres czasu, ale z kolei pozwala na utworzenie większej tablicy, wymazanie pozycji zbędnych, lub nawet na przekształcenie zwykłej tablicy przeszukiwanej liniowo lub binarnie w tablicę HASH, używając tylko prostych funkcji HASH i REHASH. Sieć działań na rysunku 2 przedstawia prosty algorytm wykonania przekształcenia [18]:



Efektywność metod transformacyjnych

Analizując poszczególne metody możemy określić ich efektywność stosowania [13, 19]. Najważniejszym parametrem, który interesuje programistę jest czas wyszukiwania danej pozycji. Ze względu na jego zależność od rodzaju maszyny, najlepszym miernikiem efektywności jest przeciętna liczba prób koniecznych do znalezienia konkretnego adresu szukanej pozycji.

Dla metod omówionych we wstępie, a więc dla klasycznego przeszukiwania całej tablicy, czy dla przeszukiwania binarnego, liczba prób koniecznych do znalezienia danej pozycji zależy od rozmiaru tablicy. Z kolei dla metod HASH efektywność zależy od wielkości współczynnika ładowania α, który jest określony jako stosunek zapełnionych już pozycji do rozmiaru całej tablicy. Dla uzyskania odpowiedniej szybkości działania przy wyszukiwaniu informacji, stosuje się tylko 75 procent możliwego zapełnienia tablicy, a więc α max = 0,75 ze względu na znaczny wzrost liczby prób po przekroczeniu tej granicy.

Przedstawione w tabeli wartości współczynników dla różnych rodzajów przeszukiwań są obliczone teoretycznie i wskazują one na wyższość metod transformacyjnych w stosunku do pozostałych. Wykonane w praktyce testy dla różnych zbiorów kluczy potwierdziły wyniki teoretyczne. Należy tutaj podkreślić fakt, że praktycznie otrzymane wyniki w dużej mierze zależą od cech charakterystycznych zbioru i od wybranej odpowiednio funkcji HASH. Dopiero wtedy zastosowanie jednej z metod REHASH w zależności od innych warunków pozwala na lepsze i szybsze określenie adresu dla podanego klucza.

Poszukiwanie	Współczynnik ładowania				Wzór teoretyczny	
	0.10	0.50	0.75	0,90		
binarne*)	5.60	7.90	8.50	8.80	log^2 (rozmiar*α)	
liniowe	1.06	1.50	2.50	5.50	$(1-\alpha/2)\ (1-\alpha)$	[6]
pseudo-losowe	1.05	1.40	1.80	2.60	$(-1/\alpha)\ log\ (1-\alpha)$	[6]
kwadratowe	1.05	1.40	1.60	2.30		
łańcuchowe	1.05	1.25	1.38	1.45	$1+\alpha/2$	[6]

*) dla tablicy o rozmiarze równym 500
α — współczynnik ładowania

Wnioski

Zaprezentowany przegląd metod transformacji odzwierciedla zasady kodowania tablic zawierających zbiory zapisów identyfikowanych przez charakterystyczne dla nich klucze (lub identyfikatory).
Przedstawione w literaturze zastosowania tej techniki wskazują na jej szerokie możliwości. Obecnie prace nad tym tematem dążą do znalezienia jak najlepszych metod transformacji i do jak najbardziej efektywnego wykorzystania pamięci przy zapisywaniu dużych zbiorów informacji. Wydaje się, że zastosowanie HASH może mieć również duże znaczenie przy wyszukiwaniu informacji z danych zbiorów poprzez różne rodzaje zapytań.

LITERATURA

[1] Peterson P.: Adressing For Random-Access Storage — IBM Journal of Research and Development, 1957

[2] Johnson J.: Indirect Chaining Method For Addressing On Secondary Keys CACM 4/1961, (218—223)

[3] Schay G., Spruth W. G.: Analysis Of a File Addressing Method, CACM 5/1962, (459—462)

[4] Schay G., Raver N.: A Method For Key-to-adress Transformation Techniques, IBM Journal Of Research And Dev., 4/1963

[5] Maurer W. D.: An Improved Hasch Code For Scatter Storage — CACM, 1/1968, (35—38)

[6] Morris R.: Scatter Storage Technique — CACM, 1/1968, (38—44)

[7] Radke E. F.: The Use Of Quadratic Residue Research, CACM 2/1970, (103—105)

[8] Bell J. R.: The Quadratic Quotient Method: A Hash Code Eliminating Secondary Clustering, CACM 2/1970, (107—109)

[9] Harrisson M. C.: Set Comparison Using Hasing Techniques AEC Resand Development Report NYO-1480-155, Courant Institute, New York Uniwersity, June, 1970

[10] Day A. C.: Full Table Quadratic Searching For Scatter Storage, CACM 8/1970, (481—482)

[11] Lampart L.: Comments On Bell's Quadratic Quotient Method For Hash Searching, CACM 8/1970, (573—574)

[12] Bell J. R., Kaman C. H.: The Linear Quotient Hash Code, CACM 11/1970, (675—677)

[13] Lin V. Y.: Key — to — address Transform Techniques, A. Fundamental Files, CACM 2/1971

[14] Karl J.: Some Properties Of Scatter Storage Technique With Linear Probing — Compute Journal 2 May, 1971, Vol. 14

[15] Lin V. Y., Yuen P. S. T.: Additional Results On Key-to-address Transform Techniques: A Fundamental Performence Study On Large Existing Formatted Files, CACM 11/1972 (996—997)

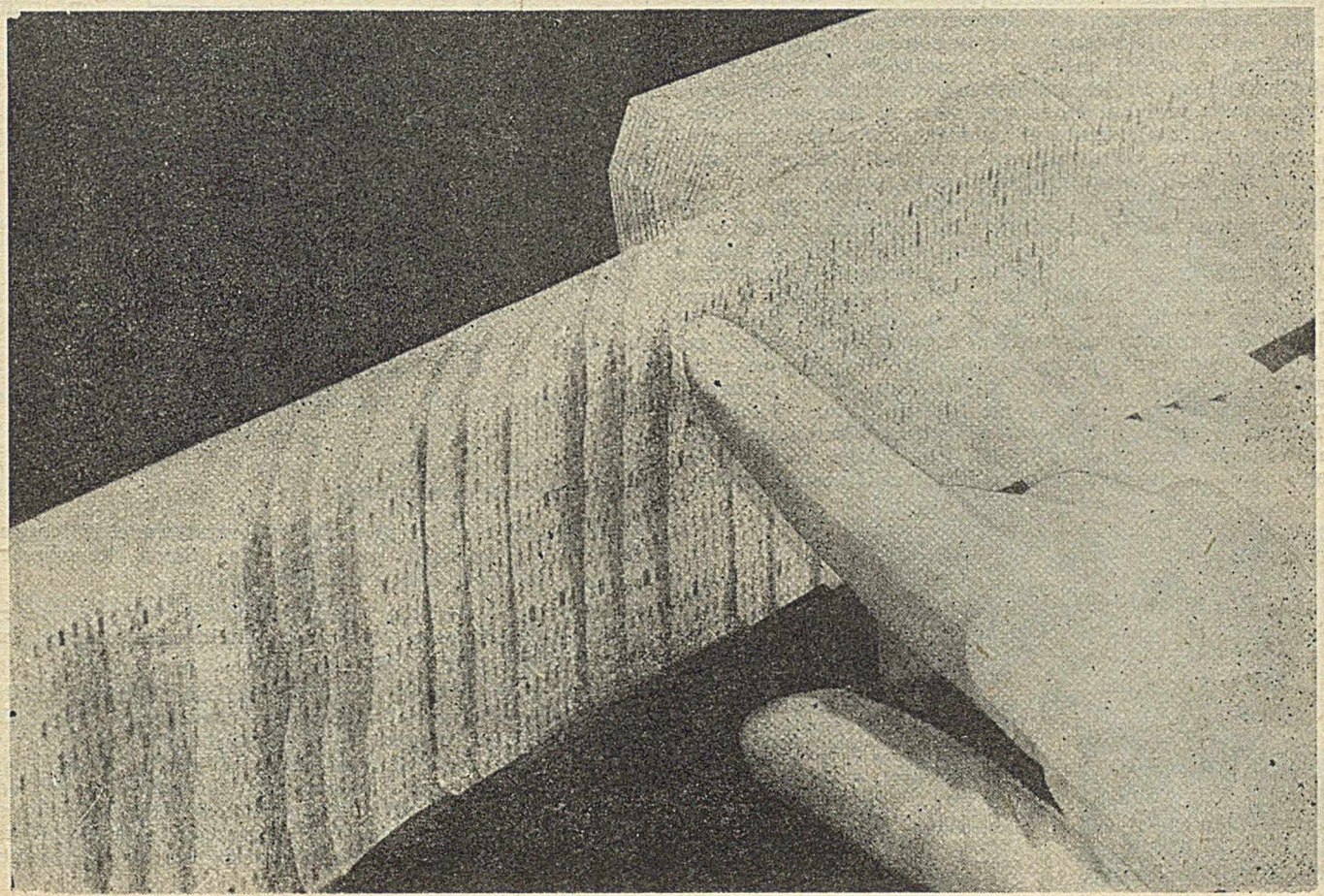
[16] Luccio F.: Weighed Increment Linear Search For Scatter Tables, CACM 12/1972, (1045)

[17] Hopgood F. R. A., Davenport J.: The Quadratic Hash Method When The Table Size Is A Power Of 2, Computer Journal 4 November, 1972, Vol. 15

[18] Bays C.: Some Techniques For Structuring Chained, Hash Tables Computer Journal 2 May, 1973, Vol. 16

[19] Dolihof T.: A Survey Of Hash Techniques, Mater. CDC. Seminarium, 1973

[20] Gurski A.: A Note Of Analysis Of Keys For Use In Hashing, BIT 1/1973, (120)



KRZYSZTOF TANNENBERG
Biuro Projektów PROSYNCHEM
Gliwice

681.327.11:621.385.832

Porównanie parametrów i możliwości komputerowych systemów graficznych

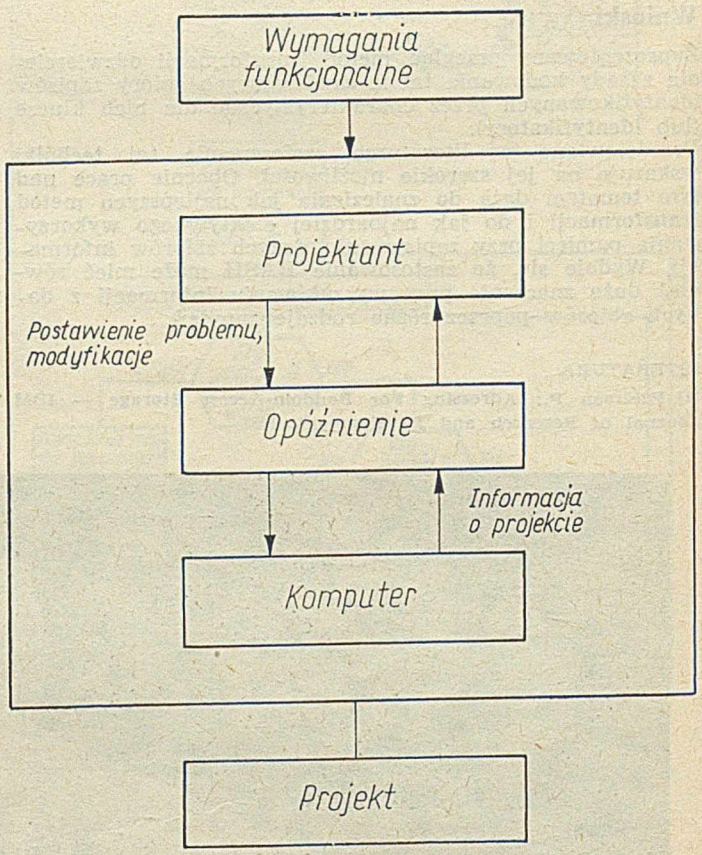
Artykuł omawia możliwości korzystania z komputerowych systemów do projektowania graficznego, przedstawiając systemy graficzne różnych producentów.

W ostatnich latach komputery stały się ważnym partnerem projektantów w procesie projektowania. Wydaje się, że spowodowane to zostało przez trzy zasadnicze czynniki:
— postęp technologiczny w produkcji komputerów sprawił, że są one coraz szybsze, bardziej niezawodne i stwarzają użytkownikowi większe możliwości zastosowania
— powstało wiele niezależnych firm software'owych, które oferują oprogramowania specjalistyczne (użytkowe) dla praktycznie każdej branży. Większość z tych firm oferuje oprogramowanie do rodzaju komputera, na jakim ma ono być eksploatowane
— do dyspozycji użytkowników oddano wiele, jakościowo nowych urządzeń we/wy (np. grafoskopy, automatyczne stoły kreślarskie, mikrofilmy, urządzenia wyjściowe COM), które ułatwiają i rozszerzają możliwości komunikacji z komputerem.

Pierwszy z wymienionych czynników spowodował, że koszt obliczeń na komputerze systematycznie maleje. Za jednego dolara można było w 1955 wykonać 10^5 instrukcji, w 1960 r. 10^6, w 1970 r. 10^8. Inaczej mówiąc, co 5 lat następuje dziesięciokrotne potanienie obliczeń [2].

Projektowanie wspomagane komputerem (*computer aided design* — CAD) jest wynikiem współpracy projektanta i komputera, w której każdy z uczestników wykonuje to co potrafi najlepiej. Projektant jest odpowiedzialny za koncepcyjny aspekt projektu tj. za wszystkie te elementy projektu, które wymagają inwencji twórczej. Zadaniem komputera jest wykonać obliczenia w najbardziej efektywny sposób i przedstawić je w najdogodniejszej formie.

Proces projektowania wspomaganego komputerem przedstawiony jest na rys. 1. Jak wynika z rysunku, podstawowym problemem w tym procesie jest opóźnienie występujące pomiędzy projektantem a komputerem. Szczególnie duże jest ono w systemach czasu nierzeczywistego, tj. przy wsadowym trybie pracy, gdzie wynosi od kilku godzin do kilku dni. Stawia to pod znakiem zapytania użyteczność tej metody w codziennej pracy projektanta. Wymieniona niedogodność sprawiła, że bardzo rozpowszechniły się konwersacyjne metody współpracy z komputerem. Punktem zwrotnym w rozwoju konwersacji człowiek-komputer było wprowadzenie pracy w podziale czasu oraz rozwiązanie problemów sterowania teletransmisją dzięki czemu projektant może mieć dostęp do komputera praktycznie w każdym miejscu i w dowolnej chwili, a oczekiwanie na odpowiedź zmniejszone zostało do kilku, a najwyżej kilkudziesięciu sekund.



Rys. Schemat procesu projektowania wspomaganego komputerem

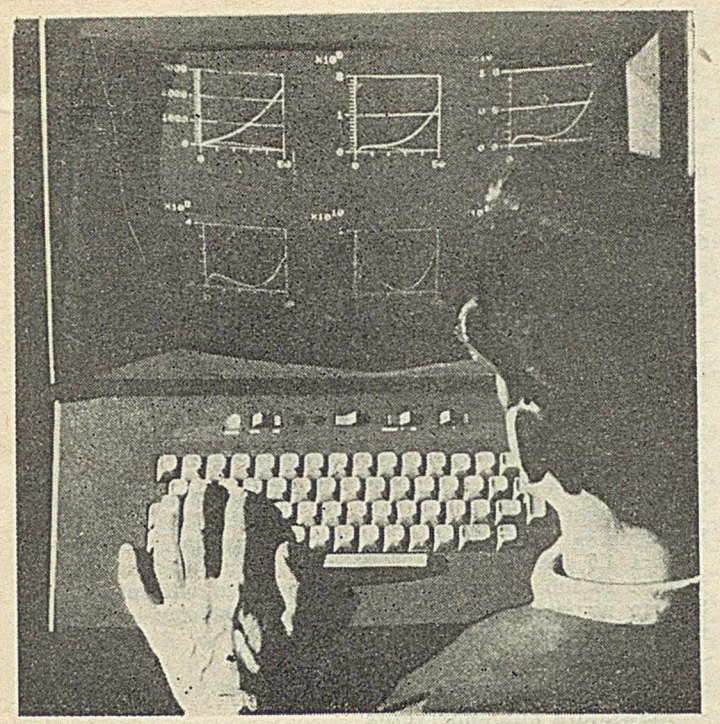
Konwersacja graficzna

Systemy graficzne są głównie wykorzystywane w konwersacyjnych systemach projektowania. Sama konwersacja realizowana może być w dwojaki sposób:

- przez przedstawienie i manipulowanie rzeczywistym rysunkiem przedmiotów (rysunki architektoniczne, części maszyn, modele cząstek itp.)
- przez wykorzystanie symboli graficznych urządzeń lub czynności (projektowanie obwodów elektrycznych, symulacja przepływów, rysunki sieci rurociągowych itp.).

Pierwszy z przedstawionych sposobów jest oczywiście bardziej skomplikowany i trudniejszy, chociażby ze względu na konieczność wykonywania rysunków trójwymiarowych, ich wymiarowania, konieczność stworzenia możliwości perspektywy i obrotów wokół osi, oraz innych manipulacji obrazem.

Mgr inż. KRZYSZTOF TANNENBERG ukończył studia na Wydziale Automatyki Politechniki Śląskiej w 1969 r. W latach 1969—73 pracował w Katedrze Elektroniki Politechniki Śląskiej. Od dwóch lat pracuje w Pracowni Informatyki w B. P. PROSYNCHEM. Autor kilku publikacji z zakresu elektronicznych układów cyfrowych.



Charakterystyki systemów graficznych

Systemy graficzne w poszczególnych firmach komputerowych

Wszystkie duże firmy komputerowe dysponują własnymi systemami graficznymi lub współpracują w tym zakresie z firmami wyspecjalizowanymi w produkcji wyświetlaczy graficznych.
Firmy komputerowe znajdujące się na polskim rynku lub zabiegające o uruchomienie swoich instalacji w Polsce proponują systemy graficzne, podane w tabeli I.
Wymienione systemy graficzne podzielić można na trzy grupy:

- systemy całkowicie zbudowane przez firmy komputerowe (np. IBM 2250, RANK XEROX — GD 101A, CDC — GRID 240)
- systemy składające się z wyświetlacza graficznego (grafoskopu) firmy elektronicznej i wyspecjalizowanego, małego komputera firmy komputerowej (np. CII — IMAGE)
- systemy, których konfiguracja ogranicza się do grafoskopu przystosowanego do współpracy z komputerem określonej firmy (TEKTRONIX 4010).

Tabela I

Nazwa firmy	CDC	CII	HONEY-WELL BULL	IBM	RANK XEROX
System komputerowy	Seria 6000 CYBER 70 CYBER 170	IRIS	Seria 6000 Seria 60	IBM 370	SIGMA
System graficzny	GRID 240	IMAGE	TEKTRONIX Seria 4010	IBM 2250	GD 101A

System graficzny GRID 240

System GRID 240 *(Series Graphic Terminal Subsystem)* jest uniwersalnym, interakcyjnym systemem graficznym.
Podstawowa konfiguracja GRID 240 składa się z jednostki centralnej i grafoskopu. Producentem systemu jest firma CDC. GRID 240 przeznaczony jest do współpracy z komputerami CDC serii 6000, CYBER 70, CYBER 170.

Elementy systemu

Zasadniczym elementem systemu jest jednostka centralna z pamięcią półprzewodnikową o pojemności 4k (lub 12k) słów 12-bitowych. Czas dostępu do pamięci wynosi 1,2 μsek. Procesor umożliwia pracę w trybie *on-line* i *off-line*. Przy pracy w trybie *on-line* jednostka centralna steruje wymianą danych pomiędzy komputerem a systemem graficznym. Podczas pracy w trybie *off-line* jednostka centralna umożliwia generację diagramów i testowanie systemu.
Ekran grafoskopu stanowi w GRID 240 21-calowa lampa kineskopowa o powierzchni użytkowej 12 × 12 cali (290 × × 290 mm).
System wyposażony jest w generator znaków i wektorów. Czas kreślenia wektora waha się w granicach od 6 do 35 μsek, punktu — od 5 do 15 μsek. Dostępne są znaki o dwóch wymiarach: 64 lub 86 znaków w wierszu. Kierunek zapisu może być poziomy lub pionowy. Czas kreślenia jednego znaku o wymiarach standardowych wynosi od 1,6 μsek do 6,4 μsek.

Urządzenia wejścia

Operator systemu GRID 240 może komunikować się z systemem poprzez: klawiaturę alfanumeryczną, klawiaturę funkcyjną, pióro świetlne.

Oprogramowanie

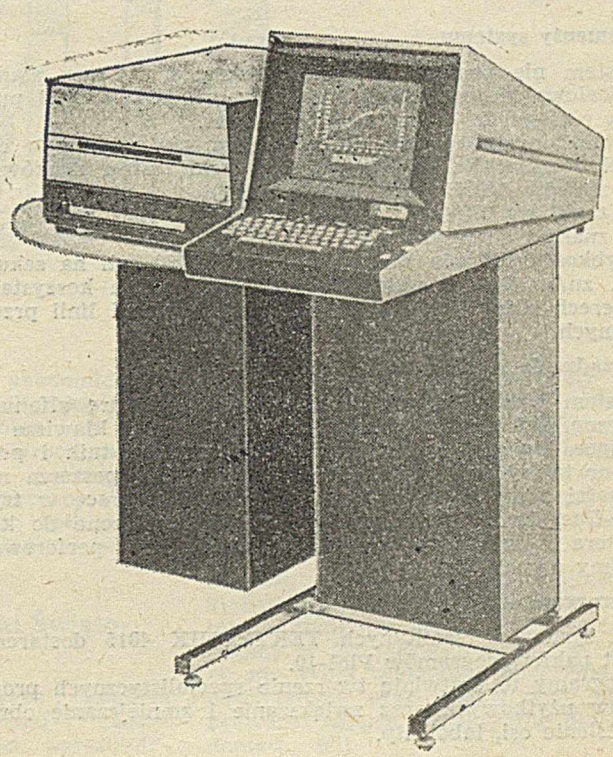
GRID 240 wspomagany jest pakietem programów GODAS *(Graphically Oriented Design and Analysis System)*. Programy napisane są w języku FORTRAN EXTENDED i mogą być wykorzystywane przy pracy konwersacyjnej i wsadowej niezależnie od tego czy terminal jest lokalny czy zdalny.
GODAS przystosowany jest do współpracy z systemami operacyjnymi KRONOS lub SCOPE/INTERCOM.

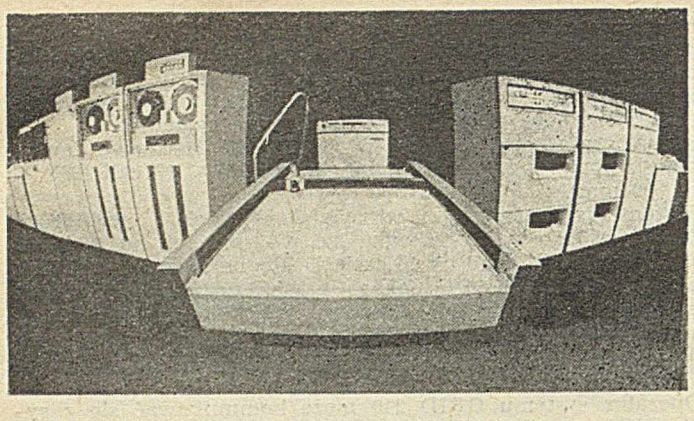
System graficzny IMAGE

System graficzny IMAGE opracowany został przez firmę CII we współpracy z Instytutem Transportowym. System oparty jest na sprzęcie firmy CII, która opracowała do tego celu specjalną wersję minikomputera MITRA 15 oraz na grafoskopie AFIGRAF firmy CSEE. System współpracuje również z systemami komputerowymi serii IRIS.

Elementy systemu

Wymianę danych i programów pomiędzy systemem graficznym a komputerem głównym organizuje minikomputer satelitarny MITRA 15/30 z pamięcią 12 k słów 16-bitowych, z 1 k słów pamięci mikroprogramowanej ROM i 64 szybkimi rejestrami.
Ekran stanowi 19-calowa lampa kineskopowa. Na ekranie mieści się 1024 × 1024 punktów. Maksymalne wymiary obrazu wynoszą 279 × 279 mm. Generator wektorów pozwala kreślić linie ciągłe, przerywane lub kropkowane, o stałej intensywności świecenia lub pulsujące. Czas wykreślenia wektora o maksymalnej długości mieści się w granicach od 1 do 128 μs.
Z generatora znaków otrzymuje się 64 znaki w kodzie ASCII. Czas zapisu jednego znaku wynosi od 4 μs do 12 μs.





Zadania urządzenia do trwałego kopiowania spełnia pisak X-Y plotter przyłączony do tego samego kabla co grafoskop. Osiągalne są wszystkie trzy rodzaje linii: ciągła, przerywana, kropkowana.

Urządzenia wejścia

Grafoskopy AFIGRAF wyposażone są w następujące urządzenia wejścia:
— klawiatura alfanumeryczna — 50 klawiszy
— klawiatura funkcyjna — 15 klawiszy programowanych
— powiększacz obrazu
— pióro świetlne.

Oprogramowanie

Grafoskopy AFIGRAF wspomagane są pakietem programów SOFA. W pakiecie tym wyróżnić można:
— instrukcje definiujące rodzaj pracy,
— instrukcje graficzne,
— instrukcje przerwań,
— instrukcje sterujące wymianą informacji pomiędzy AFIGRAF'em a komputerem.
Dostarczane przez firmę CII oprogramowanie dla systemu IMAGE rozszerza możliwości pakietu „Sofa" oraz umożliwia współpracę z systemem operacyjnym IRIS 7/8.

System graficzny TEKTRONIX seria 4010 — Model 4015

Producentem systemu jest firma TEKTRONIX. System może współpracować z komputerami wielu firm (np. IBM 370, UNIVAC, RANK XEROX, CDC, HONEYWELL BULL). Może on realizować trzy rodzaje pracy: konwersacyjny alfanumeryczny, konwersacyjny graficzny oraz wyjście graficzne z komputera.

Elementy systemu

System nie posiada własnego procesora. Ekran stanowi 19-calowa lampa kineskopowa. Ma on 1024 × 1024 punktów z czego 1024 × 780 są wyświetlane.
Generator znaków umożliwia otrzymanie znaków o czterech rozmiarach. Dostępny jest szeroki zestaw znaków:
96 znaków ASCII (duże i małe litery)
63 znaki wg TTY
88 znaków do języka APL
Szybkość kreślenia wektorów wynosi 5000 cali na sekundę 121 zn/s. Oprócz wektorów ciągłych, można korzystać z czterech różnych rodzajów linii punktowych i linii przerywanych.

Urządzenia wejścia

System wyposażony jest jedynie w klawiaturę alfanumeryczną, pokrętła do sterowania wektorami i klawisze sterujące. Do grafoskopu można przyłączyć czytnik i perforator taśmy papierowej (model 4911), który poszerza możliwości pracy systemu graficznego 4015 o pracę w trybie *off-line*. Dane z taśmy mogą być wprowadzone do komputera z szybkością do 200 zn/s. Szybkość perforowania taśmy wynosi 75 zn/s.

Oprogramowanie

Do systemów graficznych TEKTRONIX 4015 dostarczany jest pakiet programów Plot-10.
Programy te ułatwiają tworzenie specjalistycznych programów użytkowych oraz zwiększanie i zmniejszanie obrazu, kreślenie osi, tabel, itp.

System graficzny IBM 2250

System graficzny 2250 używany jest do pracy w trybie *on-line*. Umożliwia wyświetlanie, wprowadzanie zmian i manipulowanie rysunkami i danymi alfanumerycznymi. Może być wykorzystany jako konsola operatorska. Dostępny jest w dwóch modelach, które różnią się umiejscowieniem jednostki sterującej, wielkością buforu i wyposażeniem standardowym. Model 1 posiada własną jednostkę sterującą. Model 3 przyłączony jest do komputera głównego przez jednostkę sterującą 2840, która może obsługiwać do czterech systemów graficznych 2250 Model 3.

Elementy systemu

System 2250 Model 1 ma bufor o pojemności do 8 k bajtów. Bufor dla Modelu 3 mieści się w jednostce sterującej 2840, a jego pojemność dla czterech przyłączonych grafoskopów wynosi 32 k bajtów. Szybkość przesyłania informacji pomiędzy komputerem głównym a buforem wynosi 238 k bajtów dla Modelu 1 oraz 500 k bajtów dla Modelu 3.
Jako ekran wykorzystuje się 21-calową lampę kineskopową o powierzchni użytkowej 12 × 12 cali (290 × 290 mm). Dostępne są dwa rozmiary znaków: o wymiarze podstawowym i 1,5 razy większe. Czas wykreślenia jednego znaku wynosi 15 lub 17 μs w zależności od jego wielkości.
System wyposażony jest w generator wektorów bezwzględnych i przyrostowych.

Urządzenia wejścia

Urządzeniami wejścia są:
— klawiatura alfanumeryczna
— klawiatura funkcyjna — 32 klawisze
— pióro świetlne.
W systemie istnieje możliwość wykonywania trwałych kopii o wymiarach 8,5 × 11 cali (206 × 266 mm). Czas wykonywania jednej kopii wynosi ok. 20 s. Urządzenie może być sterowane zdalnie przez operatora.

Urządzenia wejściowe

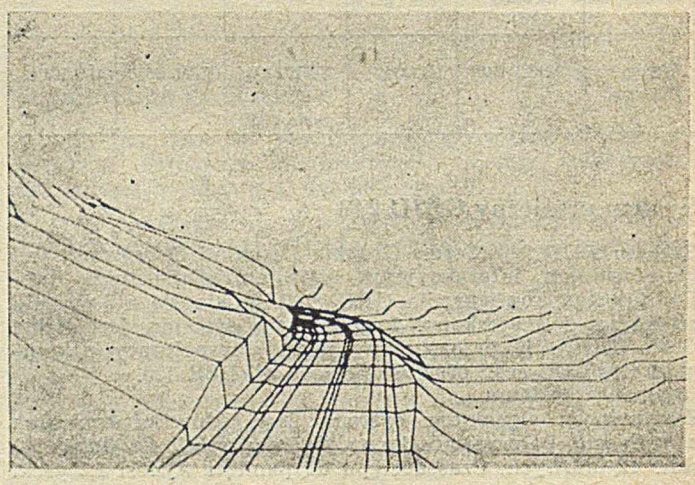
Jako urządzenia wejściowe wykorzystywane są:
— klawiatura alfanumeryczna — 70 klawiszy
— klawiatura funkcyjna — 32 klawicze
— pióro świetlne
— pióro mechaniczne.

Oprogramowanie

System graficzny GD 101 A wspomagany jest firmowym językiem do przetwarzania graficznego GL-1 Graphics Language. GL-1 utworzony jest z pakietów programów napisanych w języku FORTRAN i pracuje w komputerach firmy RANK XEROX w ramach systemu operacyjnego CP-V.

Porównanie i wnioski

Porównanie systemów graficznych oraz ocena ich przydatności w pracach projektowych jest trudna ze względu na brak jednolitych i jednoznacznych parametrów opisujących te systemy. Każda z firm podaje parametry i cechy charakterystyczne dla produkowanego przez siebie systemu, które uwypuklają jego zalety, natomiast nie podają parametrów, które mogłyby świadczyć o słabych stronach danego systemu graficznego.
Powyżej przedstawiono najważniejsze parametry systemów graficznych GRID 240, TEKTRONIX 4015, IMAGE, IBM 2250 oraz GD 101A. Oprócz parametrów podanych w



artykule należy zwrócić uwagę także na inne, trudne do ilościowego ujęcia elementy:
— możliwość manipulowania obrazem
— możliwość pracy w trybie *off-line*
— wymagana wielkość obszaru pamięci operacyjnej w komputerze głównym
— czytelność znaków i dokładność obrazu.

Z przedstawionych systemów graficznych najbardziej rozbudowaną jednostkę sterującą ma system IMAGE francuskiej firmy komputerowej CII. System ten w najmniejszym stopniu obciąża komputer główny. Pewne rodzaje obliczeń mogą być wykonywane w pętli zamkniętej przez minikomputer MITRA 15, bez angażowania komputera głównego.

W przeciwieństwie do IMAGE, system TEKTRONIX 4015 nie posiada jednostki sterującej i wszystkie potrzebne dane otrzymuje z komputera głównego. Jest to system w największym stopniu obciążający komputer główny.

Wielkości ekranów i ilości punktów wyświetlanych są we wszystkich omawianych systemach zbliżone.

Należy zwrócić uwagę na brak w systemie TETRONIX 4015 — pióra świetlnego — co znacznie ogranicza możliwości konwersacyjne tego systemu. Czynnikiem decydującym o czasie oczekiwania na odpowiedź komputera jest szybkość przesyłania informacji. Bezwzględnie najszybszą wymianę danych i programów pomiędzy komputerem głównym a systemem graficznym zapewnia IBM 2250. Szybkość przesyłu w tym systemie jest kilkakrotnie wyższa niż w GD 101 A i o dwa rzędy wyższa niż w pozostałych.

Przy pracy konwersacyjnej jest to jeden z najważniejszych parametrów.

W zakresie manipulowania obrazem, największe możliwości stwarza system GD 101 A. Także repertuar dostępnych urządzeń dodatkowych jest w tym systemie najszerszy. Dla przykładu, tylko w tym systemie możemy zmieniać intensywność świecenia wzdłuż osi Z, by osiągnąć pełniejsze złudzenie perspektywy.

Minimalna wielkość pamięci operacyjnej w komputerze głównym klasy IBM 370/145, SIGMA 6 oraz CYBER 72 potrzebna do pracy z grafoskopem wynosi 80—100 bajtów. System graficzny stanowi duże obciążenie dla komputera i przy pracy w wielodostępie może stanowić istotną przeszkodę dla innych użytkowników komputera.

Efektywna praca systemu graficznego wymaga wymiany informacji z komputerem głównym z szybkością conajmniej 9 600 bitów/s. Tak wysoka szybkość przesyłania, w wielu przypadkach, wyklucza możliwość zdalnego korzystania z komputera poprzez system graficzny i linię teletransmisyjną. Kompromis, polegający na korzystaniu z linii przesyłowych o niższych parametrach możliwy jest przy zrezygnowaniu z pracy interakcyjnej i wykorzystaniu grafoskopu jedynie jako urządzenia wyjścia.

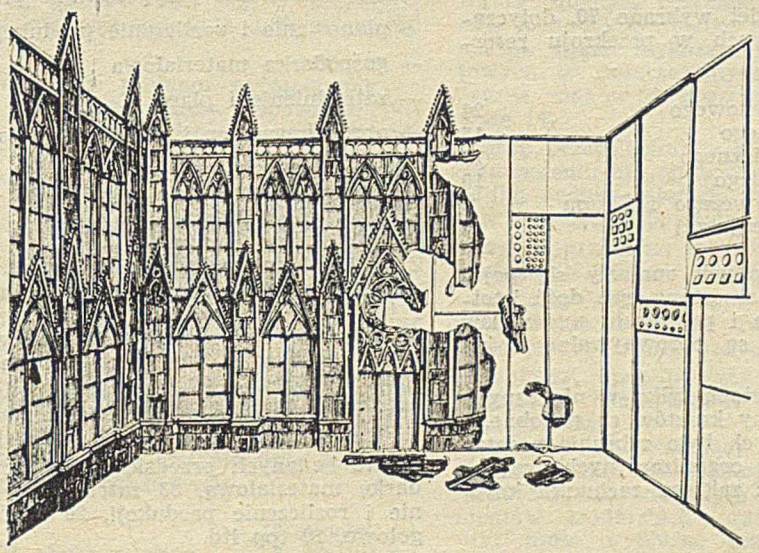
Możliwości zakupu oprogramowania specjalistycznego zaimplementowanego na przedstawionych systemach są bardzo zróżnicowane. Zdominowanie przez firmę IBM rynków światowych sprawia, że oprogramowanie specjalistyczne systemu IBM 2250 jest najłatwiej i w najszerszym stopniu dostępne.

Dotychczasowy rozwój systemów graficznych sprawił, że znalazły one zastosowanie w wielu dziedzinach projektowania i modelowania. Do dziedzin w których wykorzystywane są najczęściej zaliczamy projektowanie mostów, dróg, konstrukcji stalowych, obwodów elektrycznych, statków, siatek PERT, układów optycznych, cyfrowych układów logicznych, transformatorów, rurociągów, systemów wymiany ciepła.

LITERATURA

[1] Kopp C. G., Wertz J. B.: Designed and computer: an engineering partnership, Battele Technical Review, Ferbruary 1968.
[2] Martin J.: Design of man-computer dialogues, Prentice-Hall, Inc. Englewood Cliffs, New York, 1973.

INFORMATYKA NA UCZELNIACH



W uniwersytetach i na politechnikach

Nowym kierunkiem studiów w polskich uczelniach wyższych będzie informatyka. Kształcenie kadry informatycznej z patentem magisterskim stało się już sprawą pilną, jako że pozostajemy tu w tyle za innymi krajami socjalistycznymi.

Obowiązek kształcenia informatycznego przyjmują zarówno uniwersytety, jak i politechniki, dzieląc między siebie specjalności. Na 4,5 roczne studia w specjalności „Oprogramowanie i metody informatyki" przyjmą młodzież od nowego roku akademickiego trzy uniwersytety (Jagielloński, Warszawski i Wrocławski) oraz trzy politechniki (Warszawska, Wrocławska i Gliwicka). Wymienione politechniki, wraz z Gdańską, rozpoczynają też kształcenie studentów w specjalności „Konstrukcja sprzętu informatycznego". (elk)

ODRY dla Politechniki Świętokrzyskiej

Politechnika Świętokrzyska zawarła z Zakładem Elektronicznej Techniki Obliczeniowej w Kielcach porozumienie, na mocy którego utworzony zostanie w Radomiu wspólny ośrodek ETO wyposażony w dwie ODRY 1304. Uczelni zapewni to właściwą bazę do nauczania informatyki i rozwijania prac naukowo-badawczych, regionowi kieleckiemu — dalszy rozwój informatyki.

Ponadto Kielecki Oddział Politechniki Świętokrzyskiej wzbogacił się ostatnio o komputer ODRA 1325, zainstalowany w Zakładzie Informatyki, który nie tylko wykonuje obliczenia do uczelnianych prac badawczych, ale i prowadzi zajęcia dydaktyczne dla studentów. (el)

BOGUMIŁ STACHURA
Zakład Elektronicznej Techniki Obliczeniowej
Kielce

681.322.004.14:681.325.67:658.001

Rachunek kosztów w świetle badań ankietowych

Artykuł jest omówieniem wyników badań ankietowych przeprowadzonych w 1974 r. przez ZETO Kielce dotyczących rachunków kosztów przedsiębiorstw stosujących EPD. Brano pod uwagę przede wszystkim koszty związane z zakresem przetwarzanych informacji (z uwzględnieniem sprzętu), organizacją przetwarzania oraz tworzeniem zbiorów i wydawnictw.

Koszty były, są i będą najbardziej obserwowanym i analizowanym elementem działalności przedsiębiorstw. Szczególnego znaczenia nabierają one w obecnym okresie, kiedy ceny surowców, materiałów, maszyn i urządzeń oraz siły roboczej ciągle rosną. Chcąc analizować rachunek kosztów przedsiębiorstwa i równocześnie podejmować racjonalne decyzje, trzeba mieć szybkie i precyzyjnie sformułowane informacje o funkcjonowaniu wszystkich elementów kosztowych danej organizacji gospodarczej.

Od marca do maja 1974 r. ZETO-Kielce przeprowadziło pod patronatem naukowym SGPiS badania ankietowe dotyczące rachunku kosztów z zastosowaniem maszyn analitycznych i komputerów. W oparciu o dane Krajowego Biura Informatyki wybrano 100 przedsiębiorstw, które w przekazywanej sprawozdawczości dotyczącej „zastosowań informatyki", uwzględniły również przetwarzanie danych z zakresu rachunku kosztów.

Spośród ogółu wypełnionych ankiet wybrano 70, dotyczących przedsiębiorstw przemysłowych w przekroju resortów:

— Ministerstwo Przemysłu Maszynowego	24
— Ministerstwo Przemysłu Ciężkiego	15
— Ministerstwo Przemysłu Chemicznego	12
— Ministerstwo Przemysłu Lekkiego	10
— Ministerstwo Przemysłu Spożywczego i Skupu	5
— Ministerstwo Górnictwa i Energetyki	4

W grupie ankiet nie uwzględnionych znalazły się pojedyncze odpowiedzi z kilkunastu resortów oraz dość istotna grupa z resortu budownictwa i przemysłu materiałów budowlanych, którego dane nie są porównywalne z innymi.

Natomiast w wybranych 70 ankietach tylko w nielicznych przypadkach przetwarzane systemy kosztów są podobne.

Głównym celem badań ankietowych było zebranie podstawowych informacji o warunkach organizacyjnych związanych z przetwarzaniem danych z zakresu rachunku kosztów. Ankieta zawierała 15 pytań, które umownie podzielono na 3 grupy:

- informacje o zakresie przetwarzanych elementów rachunku kosztów oraz na jakim sprzęcie informatycznym
- informacje o organizacyjnym przygotowaniu przedsiębiorstw do wdrożenia EPD kosztów
- informacje o zbiorach w pamięci zewnętrznej komputera i wydawnictwach.

Mgr BOGUMIŁ STACHURA ukończył w roku 1966 Wyższą Szkołę Ekonomiczną w Krakowie. Od 8 lat pracuje w dziedzinie informatyki, przechodząc kolejne stopnie wtajemniczenia — od obsługi maszyn licząco-analitycznych poprzez projektowanie, kierowanie ZETO-Kielce (przez 4 lata), a ostatnio jako główny specjalista do spraw inmatyki w Ministerstwie Oświaty i Wychowania w Warszawie.

Pytania dotyczące przetwarzanych elementów rachunku kosztów

W pierwszej kolejności dokonano próby sprecyzowania terminów system i podsystem (lub element systemu) rachunku kosztów. Przyjęto następujące kryteria określające system rachunku kosztów: jednolita klasyfikacja i indeksacja, zorganizowany bank danych kosztowych *(ex ante* i *ex post)* oraz odpowedni system korzystania ze zbiorów w komputerach. Natomiast za podsystem lub element systemu uznano jeden z podsystemów wg nomenklatury GUS, w którym koszty stanowią określoną część całości.

Nazwy podsystemów przyjęto wg klasyfikacji GUS:

— tpp (techniczne przygotowanie produkcji)

— planowanie i rozliczanie produkcji

— gospodarka materiałowa

— zatrudnienie i płace

— obrót towarowy i wyrobami gotowymi

— środki trwałe

— transport i usługi

— inne

Z ankiety wynika, że na 70 odpowiedzi zaledwie w 4 przypadkach przetwarzanie kosztów nazwano systemem. Ale i tu — po przeprowadzeniu bliższej analizy — powstają wątpliwości, czy rzeczywiście przetwarzanie danych z zakresu kosztów w tych zakładach można uznać za system.

Dane z ankiety charakteryzują nie tylko przetwarzane koszty w poszczególnych podsystemach, ale mówią również o strukturze przetwarzanych podsystemów.

Na 70 badanych przedsiębiorstw aż 58 przetwarza gospodarkę materiałową, 33 zatrudnienie i płace, 31 planowanie i rozliczenie produkcji, 20 obrót towarowy i wyroby gotowe, 19 tpp itd.

Wiele przedsiębiorstw przetwarza więcej niż jeden podsystem. Jeden lub dwa podsystemy przetwarza 38 przedsiębiorstw, trzy podsystemy przetwarza 10, cztery — 12, więcej niż cztery — 11 przedsiębiorstw. Z danych zawartych w ankietach wynika, że najbardziej popularne są dwa podsystemy przetwarzane równocześnie (w 36 przypadkach), t.j. gospodarka materiałowa oraz zatrudnienie i płace.

Stosunkowo duża liczba przedsiębiorstw (18) przetwarza na maszynach licząco-analitycznych, a ponadto pewna grupa (12) przetwarza równocześnie na MLA i EMC.

Spośród maszyn cyfrowych najbardziej popularne są ODRA 1304 i 1305, które na ogólną liczbę 52 stanowią prawie połowę bo 25. Ponadto w 7 przypadkach dane kosztowe są przetwarzane na MIŃSKU 22 i w 5 przypadkach na ZAM 41.

Narzuca się tu wniosek, że w organizacji, projektowaniu i programowaniu określonego podsystemu informatycznego

dość istotną rolę odgrywa rozpowszechnienie określonego modelu lub rodziny komputerów, bowiem umożliwia ono korzystanie z jednolitych metod i technik projektowania i programowania.

Analizując zakres eksploatowanego rachunku kosztów ankietujący zastosowali podział w normatywny rachunek kosztów i planowanie kosztów jako dwie oddzielne grupy. Kierowano się tym, że nie we wszystkich przedsiębiorstwach stosuje się kompleksowy NRK (normatywny rachunek kosztów) również do planowania kosztów. Ponadto do planowania kosztów wchodzą również koszty pośrednie, określone na podstawie danych statystycznych, bądź też sporządza się tzw. „preliminarze kosztów pośrednich" w oparciu o zużycie orientacyjne. Zdecydowana większość przedsiębiorstw (62) przetwarza koszty poniesione (*ex post*). Należy jednak podkreślić, że stosunkowo duża grupa przedsiębiorstw przetwarza również normatywny rachunek kosztów, w czym przoduje przemysł maszynowy.

Pytania dotyczące organizacyjnego przygotowania przedsiębiorstw

Uzyskano odpowiedzi dotyczące przede wszystkim norm i normatywów oraz wzorów dokumentów i symboli. Normy czasowe opracowały i wdrożyły 33, a normy materiałowe — 44 przedsiębiorstwa. Tak normy czasowe, jak i normy materiałowe wiążą się ściśle z normatywnym rachunkiem kosztów (NRK). W kilkunastu przypadkach wdrożenie norm wyprzedza przetwarzanie NRK.

Z analizy ankiet wynika również, że normy czasowe i materiałowe są opracowywane i wdrażane w tych przemysłach, gdzie proces produkcyjny jest w miarę ukształtowany i sformalizowany. Najlepszym przykładem jest przemysł maszynowy, gdzie dominuje produkcja seryjna. Na 24 ankietowane przedsiębiorstwa przemysłu maszynowego aż 19 wdrożyło normy czasowe i 18 wdrożyło normy materiałowe, co stanowi odpowiednio 80% i 75%. W przemyśle ciężkim wskaźnik ten wynosi około 50%, w przemyśle chemicznym dla norm czasowych zaledwie 33%, natomiast dla norm materiałowych aż 90%. Ale już w przemyśle lekkim normy czasowe wynoszą zaledwie 10%, a normy materiałowe 70% itd.

Nieco inaczej przedstawia się problem normatywów — preliminarzy kosztów pośrednich. Liczba odpowiedzi potwierdzających (22) opracowanie i wdrożenie preliminarzy kosztów pośrednich jest zbieżna z liczbą odpowiedzi potwierdzających przetwarzanie planowania kosztów (22). Ustalanie normatywów lub preliminarzy kosztów pośrednich jest dość złożonym problemem i odpowiednio mniej przedsiębiorstw opracowało i wdrożyło to zagadnienie. Można jednak wysnuć wniosek ogólny, że przetwarzanie planowania kosztów odbywa się tam, gdzie są opracowane normy kosztów bezpośrednich i normatywy — preliminarze kosztów pośrednich.

Z głębszej analizy ankiet wynika, że szereg norm i normatywów jest opracowywane i wdrażane tylko w przedsiębiorstwach.

Odpowiedzi na niżej sprecyzowane pytanie były następujące: czy normy i normatywy opracowano

a) w przedsiębiorstwie	— 40 odpowiedzi
b) w przedsiębiorstwie i zjednoczeniu	— 5 odpowiedzi
c) tylko w zjednoczeniu	— 4 odpowiedzi
d) nie opracowano	— 21 odpowiedzi

Z danych powyższych należy wnioskować, że poważna liczba przedsiębiorstw opracowywuje normy i normatywy indywidualnie. Nie jest to właściwe rozwiązanie problemu, bowiem jednolite normy i normatywy powinny obowiązywać przynajmniej na poziomie zjednoczeń, jeśli nie na poziomie resortów. Można nawet przypuszczać, że w niektórych zjednoczeniach dwa różne przedsiębiorstwa produkujące podobne wyroby lub detale mają różne normy, nie mówiąc już o normatywach czy preliminarzach kosztów pośrednich. Istnieje więc obiektywna konieczność aby normy jednostkowe były jednolicie opracowywane na szczeblu zjednoczeń i wdrażane do przedsiębiorstw.

Następnym dość ważnym, a jednocześnie przysparzającym wiele kłopotów zagadnieniem jest problem dokumentów źródłowych. Na 70 badanych przedsiębiorstw aż w 49 zmieniono dokumenty i przystosowano je do wymogów techniki obliczeniowej. W zdecydowanej większości przedsiębiorstwa te opracowywały druki indywidualnie, dostosowując je do własnych potrzeb. Z analizy ankiet wynika, że różnorodność tych druków — nawet przy jednakowych podsystemach jest dość znaczna. Komisja Normalizacji Druków działająca przy Ministerstwie Finansów postanowiła tę sytuację unormować. W tym celu wydano pierwszy wzornik dokumentów z zakresu gospodarki materiałowej obowiązujący w całej Polsce.

Wydawało by się, że w ten sposób można rozwiązać problem, jednakże doświadczenia w tym zakresie są nieco inne. Dopóki zakres informacji na dokumentach był bardzo ograniczony (np.: dotychczasowe druki CWD), można było je wdrożyć w skali całego kraju. W sytuacji gdzie na dokumenty chcemy wytłoczyć jak najwięcej informacji źródłowych, zróżnicowanie treści tych informacji dla różnych gałęzi gospodarki narodowej będzie znaczne. Choćby przykład przemysłu maszynowego oraz górnictwa i energetyki, gdzie nawet indeks materiałowy dla tych dwu gałęzi będzie różny pod względem długości, nie mówiąc już o takich elementach, jak numer rysunku konstrukcyjnego, numer części, detalu, numer operacji technologicznej itp., które w górnictwie po prostu nie występują. Wydaje się, że druki powinny być opracowywane wg potrzeb określonych gałęzi lub branż gospodarki narodowej i w tej postaci dostosowane do aktualnych potrzeb przedsiębiorstw.

Wreszcie ostatnie zagadnienie organizacyjnego przygotowania przedsiębiorstw to indeksy i symbole z zakresu kosztów. Z analizy ankiet wynika, że wszędzie tam gdzie wdrożono ewidencję kosztów (62) zmieniono również symbolikę kosztów (62).

Na symbolikę kosztów zwrócono w ankiecie szczególną uwagę, poświęcając jej sporo miejsca. Ujednolicenie symboliki w przedsiębiorstwach ma bardzo istotny wpływ na projektowanie systemów informatycznych oraz ich późniejsze przetwarzanie. Jednolita i strukturalnie prosta symbolika ułatwia również programowanie, a w końcowym efekcie umożliwia w tym zakresie optymalną pracę elektronicznej maszyny cyfrowej.

Na 70 badanych przedsiębiorstw, 62 odpowiedziało, że stosuje jednolitą symbolikę w całym przedsiębiorstwie. Dalsza analiza ankiet wykazała jednak, że długość symboli kosztowych w poszczególnych przedsiębiorstwach jest różna. Liczba znaków w poszczególnych symbolach waha się od 3 do 14.

Szereg przedsiębiorstw podległych organizacyjnie jednemu zjednoczeniu ma opracowaną i wdrożoną niejednolitą symbolikę. Jednak zaledwie w 13 przypadkach symbolika kosztów jest jednolita w branży (zjednoczeniu).

Jakie są przyczyny tak różnicowanych struktur kosztowych i długości symboli? Należałoby do nich zaliczyć:

— autorzy opracowujący indeksy kosztowe (księgowi-kosztowcy) często nie uwzględniają warunków elektronicznego przetwarzania danych

— autorzy, opracowujący indeksy kosztowe (informatycy) przyjmują bezkrytycznie wszystkie życzenia księgowych-kosztowców

— autorzy opracowujący indeksy kosztowe często zapominają o tym, że wielkość i złożoność symboliki kosztów może stać się poważnym hamulcem w wypełnianiu dokumentów źródłowych przez pracowników przedsiębiorstw oraz może wydłużać czas pracy komputera. Z analizy tej można wyprowadzić wniosek, że symbolika kosztów winna być opracowywana przez fachowców z zakresu rachunku kosztów, którzy uwzględniają zarówno warunki EPD, jak i niezbędny zakres informacji wynikający z przepisów i potrzeb zarządzania przedsiębiorstwem oraz możliwości szybkiego i sprawnego wypełniania dokumentów. Należy tu dodać, że w Centralnym Urzędzie Geologii wszytkie przedsiębiorstwa wchodzące w skład tego resortu w ciągu 1975 r. będą miały obowiązek wdrożyć jednolitą symbolikę kosztów, jednolite dokumenty źródłowe oraz wiele innych jednolitych kodów. W ankiecie zadano również pytanie, czy symbolika kosztów jest warunkiem koniecznym w przetwarzaniu kosztów i czy winna obejmować wszystkie koszty przedsiębiorstwa. Na 70 ankietowanych 60 odpowiedziało, że w warunkach przetwarzania danych z zakresu rachunku kosztów opracowanie i wdrożenie symboliki kosztów jest warunkiem koniecznym. Nieco mniej, bo 55 odpowiedzi mówi, że symbolika kosztów winna obejmować wszystkie koszty przedsiębiorstwa. Liczba

odpowiedzi negatywnych, choć niezbyt duża, świadczy o niedocenianiu tych problemów. Czasem poprostu symboli kosztów nie zmienia się, a jedynie przystosowanie do wymogów EPD.

Taki stan rzeczy nie tylko utrudnia prace projektowo-programowe i eksploatacyjne, ale przede wszystkim uniemożliwia integrację kosztów w skali zjednoczenia.

Ostatnie pytanie z tej grupy dotyczy powiązania symboliki kosztów (ściśle, dowolnie) z typowymi lub branżowymi planami kont.

Na pytanie czy symbolika winna być ściśle czy dowolnie powiązana z typowymi lub branżowymi planami kont odpowiedziało 45 ankietowanych, 16 odpowiedziało, że może być dowolnie powiązana, natomiast 9 nie ma zdania.

Należy przypuszczać, że gdyby ankietę wypełniali pracownicy pionu księgowości odpowiedź byłaby jednoznaczna tzn. wszyscy opowiedzieliby się za ścisłym powiązaniem z planami kont. Ścisłe powiązanie symboliki kosztów z branżowymi lub typowymi planami kont ma bardzo istotny wpływ na pełną automatyzację nie tylko kosztów, ale całej księgowości przedsiębiorstw. Ścisłe powiązanie umożliwia również uzyskiwanie odpowiedzi w przekroju kosztów do kont syntetycznych, analitycznych i układu rodzajowego. Ponadto zabezpiecza ona dalsze, bardziej szczegółowe rozbicie kosztów, co jest wręcz niezbędne do analizy kosztów poniesionych.

Pytania dotyczące informacji o zbiorach w pamięci zewnętrznej i wydawnictwach komputerów

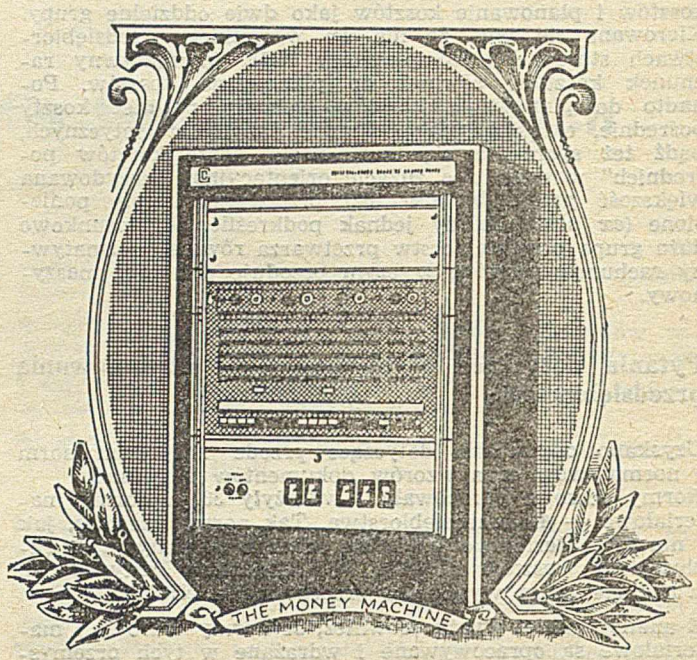
Na 70 odpowiedzi w 23 przypadkach poinformowano, że dane do przetwarzania kosztów uzyskuje się tylko z dokumentów źródłowych. Są to przedsiębiorstwa, które przetwarzają na maszynach licząco-analitycznych (18) oraz te, które zbiory traktują przejściowo, kasując je po każdych obliczeniach. Druga grupa badanych jest bardziej interesująca, bowiem dane do przetwarzania kosztów uzyskuje się tam z dokumentów źródłowych i ze zbiorów w pamięci zewnętrznej.

Z doświadczeń w przetwarzaniu wynika, że w wielu systemach czy podsystemach informatycznych tworzy się dwa podstawowe zbiory: zbiór zmienny i zbiór stały. Zbiór zmienny zabezpiecza dopływ nowych informacji i aktualizuje poprzednie oraz przechowuje informacje niezbędne do analiz długookresowych. Natomiast zbiór stały zabezpiecza podstawowe parametry kosztowe wraz z cenami, stawkami, procentami, wskaźnikami itp. Zbiór stały winien być systematycznie aktualizowany i w miarę reorganizowany, aby usprawniać metody dostępu do informacji. Zdecydowana większość badanych bo aż 90% przedsiębiorstw (jeśli od ogólnej liczby 70 odjąć 18, które przetwarzają dane kosztowe na maszynach licząco-analitycznych), stwierdziła, że przetwarzanie kosztów opiera na dokumentach źródłowych i zbiorach w pamięci zewnętrznej. Natomiast tylko 1 przedsiębiorstwo odpowiedziało, że dane do przetwarzania kosztów otrzymuje ze zbiorów w pamięci zewnętrznej i to tylko dla NRK. Ogromne trudności w sklasyfikowaniu i przedstawieniu w odpowiedniej postaci nastręczało zagadnienie zbiorów w pamięci zewnętrznej.

Istnieje w tym zakresie w Polsce dość znaczna różnorodność pojęć, terminów, zasad organizacji zbiorów, ich zakresu merytorycznego itp. Dotychczas nie dokonano nawet prób ustalenia choćby ramowych standardów zbiorów pamięci zewnętrznej, które określałyby zawartość głównych zagadnień przetwarzanych podsystemów informatycznych. Zagadnienie to jest bardzo istotne dla projektanta i programisty systemów EPD. Szereg przedsiębiorstw przedstawiło nazwy zbiorów, które niewątpliwie świadczą o wysokim poziomie organizacyjnym, ale też wskazują na olbrzymie zróżnicowanie pojęć nawet w ramach jednej gałęzi przemysłu.

Na przykład w przemyśle maszynowym dokonano klasyfikacji zbiorów kosztowych wg następujących kryteriów:
a) zbiory z zakresu gospodarki materiałowej
b) zbiory z zakresu robocizny i pracochłonności
c) zbiory wspólne dla kosztów normatywnych
d) zbiory pomocnicze.

W otrzymanym z przedsiębiorstw materiale badawczym dała się zauważyć różnorodność treści i nazw zbiorów. Często przewijały się takie pojęcia jak: zbiór norm jednostkowych (jakich?), kartoteka konstrukcyjno-technologiczna (co zawiera?), kartoteka strukturalna lub kartoteka zwinięć i rozwinięć wyrobów itp. Z punktu widzenia treści zbiory te są do siebie bardzo podobne, ale mają różne nazwy. Różnie też potraktowano zagadnienie zbiorów stałych i zmiennych.



Brak jest jednoznacznych określeń, jaki zbiór należy uznać za stały, a jaki za zmienny. Czy kryterium tym ma być ilość i częstotliwość aktualizacji, czy jeszcze inne kryteria? Czy normy jednostkowe wprowadzone do pamięci zewnętrznej,np. w rozmiarze 10 000 zapisów z miesięczną aktualizacją w liczbie około 500 zmian należy uznać za zbiór stały? Jakim wobec tego będzie zbiór stanów materiałowych (np. 10.000 zapisów) z miesięczną aktualizacją w liczbie około 5 000 zmian?

Jeśli chodzi o problem wydawnictw, to i w tym zakresie istnieje ogromna różnorodność tak w treści, jak i w nazwach. Wystarczy powiedzieć, że jedno z przedsiębiorstw wykonuje miesięcznie na komputerze około 400 różnych tytułów tabulogramów (nie tylko kosztownych). Bardzo często w wielu przedsiębiorstwach tej samej gałęzi przemysłu czy nawet zjednoczenia wykonuje się identyczne co do treści tabulogramy, ale różne w nazwie i budowie wewnętrznej. Znów określenie pewnego standardu tabulogramów tak w nazewnictwie, jak i treści, choćby tylko dla zjednoczeń, ogromnie usprawniłoby prace projektowo-programowe. Podobnie, jak w symbolice kosztów, tak i dla zbiorów w pamięci zewnętrznej oraz tabulogramach postanowiono stworzyć w przedsiębiorstwach Centralnego Urzędu Geologii jednolite struktury treści i tytuły. Aktualnie już w czterech przedsiębiorstwach wdrożono te zasady w zakresie EPD gospodarki materiałowej i przedmiotów nietrwałych.

*
* *

Dzięki omówionym badaniom ankietowym nie stwierdzono aby obecnie istniał w pełni zautomatyzowany system (podsystem) przetwarzania danych z zakresu rachunku kosztów, funkcjonujący na określonej elektronicznej maszynie cyfrowej wg kryteriów przyjętych na wstępie, mimo, że kilka przedsiębiorstw zaawansowało bardzo poważnie prace organizacyjne dotyczące przygotowania przedsiębiorstw do wdrożenia komputeryzacji na odcinku rachunku kosztów. Wyciągnięto również wniosek, że istnieje wielka różnorodność symboliki kosztów jak również dokumentów źródłowych. Wreszcie ankieta zasygnalizowała dość istotne problemy organizacji zbiorów w pamięciach zewnętrznych oraz treści wydawnictw. Zdaniem autora należy podjąć prace badawcze nad zagadnieniem standaryzacji w zakresie treści i tytułów w problematyce zbiorów oraz wydawnictw. Nie mniej ważnym problemem jest sprecyzowanie typowego pakietu zakładania i funkcjonowania zbiorów w pamięci zewnętrznej.

HELENA DRYZEK
Instytut Organizacji i Kierowania PAN i MNSzkWiT
Warszawa

681.322.004.14:651.8:681.327.2.002.513.5

SOCRATE—system zarządzania bazami danych

Opis systemu SOCRATE, do zarządzania bazami danych i do wyszukiwania informacji. Podano również przykłady praktycznego zastosowania systemu.

System zarządzania bazami danych SOCRATE powstał w Instytucie Matematyki Stosowanej w Grenoble a w obecnej wersji jest rozpowszechniany przez CII (Compagnie Internationale pour l'Informatique). Istniejące wersje systemu są przystosowane do komputerów CII IRIS lub SIEMENS (różne modele). Opracowano wersję systemu SOCRATE na maszynę IBM, ale dotychczas nie jest ona wykorzystywana.

System SOCRATE pozwala definiować bazy danych, a następnie dokonywać na nich różnego rodzaju operacji dopuszczając wiele sposobów dostępu do danych. System pozwala na pracę w języku SOCRATE (tryb wsadowy lub konwersacyjny) lub w języku wyższego poziomu. SOCRATE jest systemem wielodostępnym.

Na uwagę zasługuje oryginalne rozwiązanie problemu organizacji pamięci, pozwalające użytkownikowi swobodnie dysponować pamięcią wirtualną w trakcie definicji bazy danych. Baza danych ma charakter dynamiczny i może ulegać modyfikacji przez dodanie lub usunięcie danych.

Funkcje systemu SOCRATE

SOCRATE jest systemem zarządzania bazami danych dysponującym językiem przeznaczonym do definicji bazy danych oraz językiem programowania [1] pozwalającym dokonywać różnego rodzaju operacji na danych. W systemie przewidziano możliwość realizacji programów i makrorozkazów zdefiniowanych przez użytkownika.

W systemie SOCRATE przewidziano trojakiego rodzaju tryb pracy:

— wsadowy (język SOCRATE)
— konwersacyjny (język SOCRATE)
— interfejs w języku wyższego poziomu (SOCRATE/COBOL)

Istotne różnice między SOCRATE i SOCRATE/COBOL są następujące:

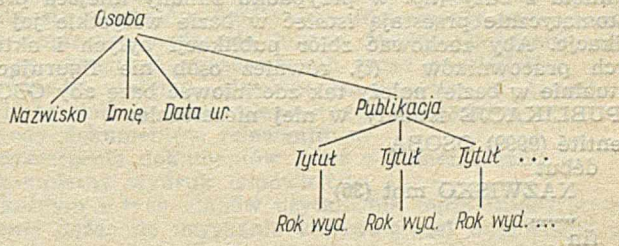
- SOCRATE pozwala wprowadzać dane tylko z dysku a SOCRATE/COBOL również z taśmy magnetycznej
- w systemie SOCRATE można zdefiniować wiele baz danych (do 100), ale możliwość jednoczesnego dostępu ograniczona jest tylko do jednej bazy (w trybie konwersacyjnym możliwy jest jednoczesny dostęp do wielu baz, ale pod warunkiem korzystania z wielu końcówek, przy czym dostęp z jednej końcówki jest ograniczony do jednej bazy danych; oczywiście możliwy jest dostęp do jednej bazy zrealizowany jednocześnie z wielu końcówek). SOCRATE/COBOL pozwala na jednoczesny dostęp do wielu baz danych
- SOCRATE/COBOL działa na ogół szybciej niż SOCRATE.

Użytkownik systemu SOCRATE przystępując do definicji bazy danych dysponuje tzw. przestrzenią wirtualną wystarczająco dużą, aby pomieścić wszystkie aktualne i przewidywane dane. Do opisu elementów bazy danych (nazw elementów i ich charakterystyk) oraz istniejących między nimi związków służy język definicji bazy danych. Język ten pozwala również zdefiniować rodzaj dostępu do poszczególnych danych (dostęp bezpośredni, sekwencyjny, inwersyjny lub z wykorzystaniem odsyłaczy).

Na podstawie zadeklarowanej przez użytkownika struktury bazy, opisanej w języku definicji bazy danych, system SOCRATE rezerwuje odpowiedni obszar w pamięci, wykorzystując od tego celu mechanizm rzutowania przestrzeni wirtualnej na przestrzeń rzeczywistą (pamięć operacyjna i pomocnicza).

System SOCRATE zawiera makrogenerator, pozwalający użytkownikowi sformułować na bazie języka SOCRATE prosty język przeznaczony do specjalnych zastosowań.

Na podkreślenie zasługuje fakt, że system SOCRATE, przy swoich szerokich możliwościach, wymaga bardzo niewiele miejsca w pamięci operacyjnej. Rys. 2 ilustruje podział pamięci operacyjnej (98 Kb) [2] w przypadku systemu SOCRATE pracującego w trybie wsadowym (tryb konwersacyjny wymaga dodatkowo ok. 1 Kb pamięci dla każdej końcówki) oraz w przypadku SOCRATE/COBOL.



Znaczna różnica w rozmiarze SOCRATE i SOCRATE/COBOL wynika z tego, że SOCRATE/COBOL zawiera tylko część systemu potrzebną do realizacji programów uprzednio skompilowanych, podczas gdy SOCRATE zawiera kompilator, makrogenerator, procedury definicji bazy danych itp.

1) Możliwości SOCRATE jako języka programowania są w istocie ograniczone, zwłaszcza w zakresie operacji numerycznych. Jest to zresztą usprawiedliwione, gdyż SOCRATE jest z definicji systemem zarządzania bazami danych, a język SOCRATE przeznaczony jest do pisania programów służących temu celowi

2) Wersja SOCRATE/IBM wymaga tylko 50 Kb pamięci operacyjnej (wraz z polem roboczym: 65—80 Kb). 1 Kb = 1024 bajty

Mgr HELENA DRYZEK studia ukończyła na Wydziale Matematyki UW, sekcja Metod Numerycznych (1969 r.), pracowała w Centrum Obliczeniowym PAN, a obecnie w Instytucie Organizacji i Kierowania PAN i MNSzWiT. Zajmuje się organizacją pamięci wirtualnych (2 publikacje o charakterze monograficznym wydane w PWN, seria Prace COPAN) a od niedawna problemami wyszukiwania informacji. Odbyła staż w Sécurité Sociale w Grenoble.

Język definicji bazy danych

W języku definicji bazy danych głównym pojęciem są **obiekty** (franc. entité) będące elementami zbiorów należących do bazy. Obiekty opisane są za pomocą charakterystyk zwanych **danymi**. Przykładem obiektów są np. książki w zbiorze książek. Danymi mogą być w tym przypadku tytuł, nazwisko autora, rok wydania.

Język definicji w systemie SOCRATE służy do:
— opisu obiektów przez nadanie im nazw i określenie ich charakteru (typ danych),

— opisu związków zachodzących między zbiorami obiektów (struktura drzewa lub graf).

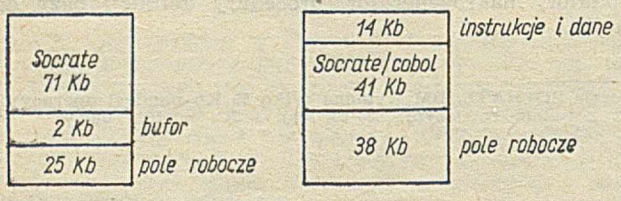
Dane można definiować jako słowo, tekst, liczbę lub listę wartości.

W strukturze bazy różne obiekty mogą być defniowane niezależnie (tj. na tym samym poziomie) lub w obrębie obiektu wyższego poziomu. Dla wyjaśnienia różnicy rozważmy przykład bazy danych zawierającej informacje o pracownikach naukowych i ich publikacjach. Informacje opisujące obiekt zawarte są w bloku ograniczonym słowami début (początek) i fin (koniec). We wszystkich przykładach słowa zastrzeżone języka SOCRATE pisane są małymi literami

```
entité (9999) OSOBA
  début
    NAZWISKO mot (30)
    IMIĘ mot (10)
    DATUR de 0 á 999999
    étite
    entité (99) PUBLIKACJA
      début
        TYTUŁ mot (30)
        ROKWYD de 1945 á 2000
fin
```

Zdefiniowana powyżej baza danych ma następującą strukturę podaną na rys. 1. Przy takiej definicji bazy danych obiekty zwane PUBLIKACJAMI istnieją wyłącznie wewnątrz obiektów zwanych OSOBAMI. Konsekwencje tego są następujące: w momencie, gdy określona osoba zostaje usunięta z bazy (np. w przypadku zmiany miejsca pracy) automatycznie przestają istnieć w bazie wszystkie jej publikacje. Aby zachować zbiór publikacji byłych i aktualnych pracowników (tj. również osób nie figurujących aktualnie w bazie) należy tak zdefiniować bazę aby OSOBY i PUBLIKACJE istniały w niej niezależnie:

```
entité (9999) OSOBA
  début
    NAZWISKO mot (30)
    .......
  fin
entité (99999999) PUBLIKACJA
  début
    AUTOR mot (30)
    .......
fin
```



Język definicji bazy danych służy także do deklaracji niektórych typów dostępu do danych i obiektów.
Określenie liczby obiektów i typów poszczególnych danych pozwala zdefiniować przestrzeń wirtualną jaką dysponuje użytkownik bazy. W momencie definicji struktury bazy monitor systemu rezerwuje odpowiednią liczbę miejsc w pamięci wirtualnej oraz w specjalnej tablicy umieszcza nazwy poszczególnych obiektów i danych wraz z odsyłaczami opisującymi istniejące między nimi związki. Deklarowana liczba obiektów jest maksymalną liczbą przewidywaną — w ten sposób zapewnia się wystarczająco dużą przestrzeń wirtualną. Dla każdego zbioru obiektów, generowany jest tzw. łańcuch bitów obecności (wyzerowany w momencie definicji struktury bazy), w którym jedynki na odpowiednich pozycjach wskazywać będą obecność poszczególnych obiektów w bazie co przyspieszy ich odszukanie.

Język SOCRATE

Język SOCRATE jest bardzo zbliżony do naturalnego języka francuskiego. Poniżej podano przykłady kilku wyrażeń w tym języku
pour ... fin — pętla typu dla
si ... alors ... fin — instrukcja warunkowa jeśli ... to ...
faire .. fin — pętla typu wykonaj ...
ayant ...; — mający (filtr używany przy obiekcie)
un (une) — jeden (jedna)
tout (toute) — każdy (każda)
Kwantyfikator un (une) służy do wskazania jednego (wybranego lub pierwszego) obiektu o danej nazwie. Kwantyfikator tout (toute) wskazuje wszystkie obiekty o danej nazwie.
Użytkownik dysponuje następującymi zmiennymi
x_i — adres wirtualny, i = 1,2, ... , 9
y_i — zmienna numeryczna, i = 1,2, ... , 10
z_i — zmienna alfanumeryczna, i = 1,2, ... , 5
W języku wyróżniono cztery główne typy operacji
inicjacja (g — géneration)
konsultacja (i — interrogation)
aktualizacja (m — mise á jour)
likwidacja (s — suppression).

Opis czterech podstawowych operacji języka SOCRATE
Inicjacja

W momencie definicji struktury bazy dla każdego zbioru obiektów, rezerwowany jest łańcuch bitów obecności i długości równej maksymalnej liczbie obiektów. Inicjacja oznacza nadanie wartości 1 bitowi wskazującemu obecność danego obiektu np. instrukcja:
g une OSOBA x_1
oznacza inicjację obiektu OSOBA o adresie wirtualnym umieszczonym w rejestrze x_1

Aktualizacja

Aktualizacja oznacza nadanie konkretnej wartości zmiennym języka lub danym zdefiniowanym w bazie np.:
m NAZWISKO de une OSOBA x_1 = 'KOWALSKI'

Konsultacja

Konsultacja oznacza zadanie pytania dotyczącego przechowywanych w bazie danych np.:

```
pour toute OSOBA
  i NAZWISKO
fin
```

Wykonanie tej instrukcji pozwoli otrzymać listę nazwisk wszystkich osób w bazie.

Likwidacja

Operacja likwidacji może być wykonywana na zmiennych języka lub bezpośrednio na obiektach bazy. W przypadku obiektu, wykonanie tej operacji powoduje usunięcie wszystkich danych związanych z obiektem oraz wyzerowanie odpowiedniego bitu w łańcuchu bitów obecności.

Dostęp do danych

W języku SOCRATE istnieją następujące typy dostępu do danych
— sekwencyjny
— bezpośredni (z użyciem numeru porządkowego lub słownika)
— inwersyjny
— odsyłacze proste lub typu pierścień.
Dostęp sekwencyjny realizowany jest do kolejnych obiektów w bazie (na podstawie łańcucha bitów obecności) z dodatkiem ewentualnych filtrów ze względu na poszczególne dane. Jeśli obiekty w bazie mogą być identyfikowane za pomocą numerów porządkowych, to powolny dostęp sekwencyjny można zastąpić szybkim dostępem bezpośrednim realizowanym do wskazanego obiektu.
Ponadto w języku SOCRATE istnieje możliwość definicji słów kluczowych będących elementami słownika, co pozwala na dostęp bezpośredni z użyciem słownika. Istnienie słów kluczowych zaznaczane jest w bazie (clé unique). Można również zdefiniować słownik synonimów definiując w bazie słowa kluczowe z odsyłaczami wprzód i wstecz (clé double).
Dostęp inwersyjny wykonywany jest do podzbioru obiektów identyfikowanych za pomocą łańcucha bitów inwersyjnych i deklarowany jest w bazie jako obiekt.
Do określenia szczególnego rodzaju związków między obiektami służą odsyłacze (franc. references), zastosowanie ich wyjaśnimy na przykładzie.
W podanym wcześniej przykładzie bazy, PUBLIKACJE zostały początkowo zdefiniowane jako obiekty niższego po-

ziomu w obrębie obiektu OSOBA. Tego rodzaju związki między obiektami powodują, że obiekty niższego poziomu istnieją zależnie od obiektów wyższego poziomu. W zmodyfikowanej wersji bazy, PUBLIKACJE zostały zdefiniowane jako obiekty tego samego poziomu co OSOBY. W tym przypadku, aby zidentyfikować publikację należy oprócz tytułu podać również nazwisko autora. Odsyłacze pozwolą w naturalny sposób określić związki między osobami i publikacjami.

Ponieważ jedna osoba może posiadać wiele publikacji, wygodnie jest zadeklarować je jako pierścień (franc. anneau). Ostatecznie zmodyfikowana baza ma następującą strukturę

```
entité (9999) OSOBA
   début
      NAZWISKO mot (30)
      .....
      DZIEŁA anneau
   fin
entité (99999999) PUBLIKACJA
   début
      TYTUŁ mot (30)
      AUTOR refere DZIEŁA de une OSOBA
   fin
```

Deklaracja „DZIEŁA anneau" powoduje rezerwację dodatkowych dwóch słów, przeznaczonych na odsyłacze, przy każdym obiekcie OSOBA. Dana zwana DZIEŁA wskazywać będzie początek pierścienia. Analogicznie, deklaracja „AUTOR refere DZIEŁA de une OSOBA" powoduje rezerwację dwóch słów (odsyłaczy) przy każdym obiekcie PUBLIKACJA. Wykonanie operacji typu aktualizacja spowoduje automatyczne określenie odpowiednich odsyłaczy. (Zauważmy, że w przypadku łańcuchów inwersyjnych użytkownik musiał sam je generować).

Przykłady zastosowań systemu SOCRATE

W 1974 roku było we Francji ponad 20 użytkowników SOCRATE — omówimy trzy przykłady zastosowań systemu.

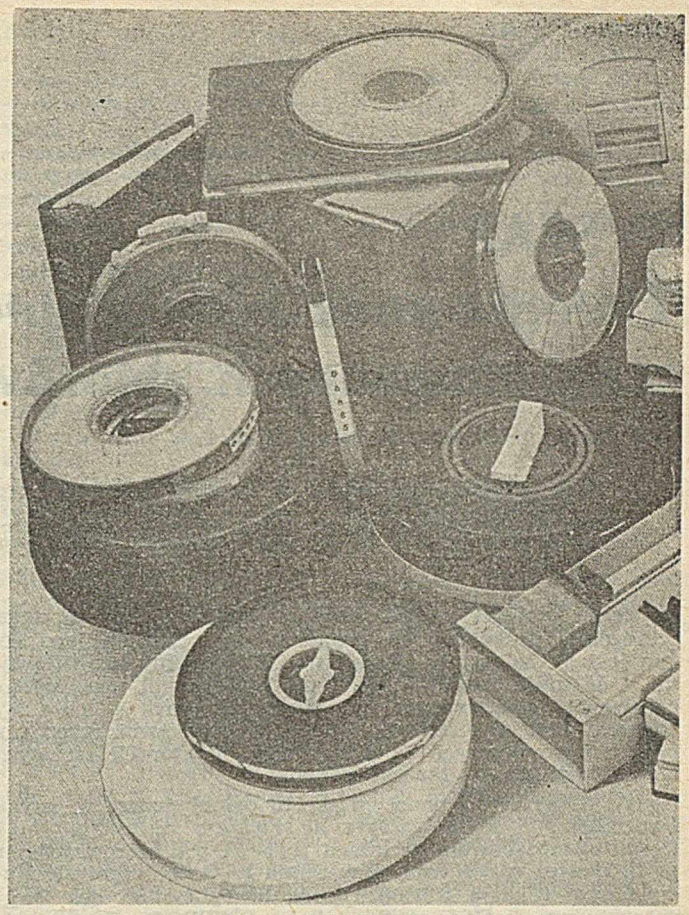
● **W szpitalu du Sablon w Grenoble** system SOCRATE jest wykorzystywany do celów administracyjnych (rejestracja chorych i zarządzanie pulą łóżek), do gromadzenia dokumentacji medycznej (karty chorób) oraz do eksploatacji bazy danych z zakresu toksykologii.

Czynności administracyjne wykonuje system SIGAL (Systéme Informatique de Gestion des Lits) działający w trybie natychmiastowym. Szpital liczy 3000 łóżek, dziennie przyjmuje się ok. 150 pacjentów. W momencie przyjmowania nowego pacjenta, obsługa systemu informuje się, za pośrednictwem konsoli, o numerze wolnego łóżka na wskazanym oddziale i dokonuje z konsoli rejestracji pacjenta. SIGAL udziela informacji o stanie wykorzystania łóżek na poszczególnych oddziałach, identyfikuje poszczególnych pacjentów oraz wystawia rachunki za leczenie. Dokumentację medyczną tworzy zbiór liczący 55 000 kart chorób zawierających dane personalne pacjenta, diagnozę, wyniki badań, listę interwencji medycznych. Czas wyszukiwania w tym systemie wynosi 51 s, wydruk odpowiedzi — 20 s. Wprowadzane do systemu karty chorób są przygotowywane przez medyczny personel pomocniczy.

Baza danych z toksykologii służy do stawiania diagnozy w przypadkach zatruć. System działa w trybie konwersacyjnym: użytkownik udziela odpowiedzi na pytania systemu o poszczególne objawy i uzyskuje listę związków toksycznych uszeregowanych wg malejącego prawdopodobieństwa zażycia. Po dostarczeniu pierwszej takiej listy, system podaje wykaz analiz, uszeregowanych wg priorytetów, jakich należy dokonać w celu uzyskania trafniejszej diagnozy, a po ich uzyskaniu dostarcza liste zmodyfikowaną. Postępowanie to można powtarzać aż do uzyskania odpowiedzi systemu, że na podstawie dostarczonych informacji nie może podać dokładniejszej diagnozy. System stawia diagnozę przy założeniu, że informacje o objawach i wyniki analiz pochodzą z 6 pierwszych godzin po zatruciu (granica efektywności systemu: 12 godzin). Baza danych jest wykorzystywana za pośrednictwem telefonu przez lekarzy z całej Francji.

Konfiguracja wykorzystywanego sprzętu: maszyna IRIS 50, wyposażona w 6 monitorów drukujących, 2 monitory ekranowe, 8 jednostek pamięci dyskowej i 6 jednostek pamięci taśmowej oraz system przygotowywania danych IBM 3741.

● **W Archiwum Paryskim** SOCRATE jest stosowany jako system wyszukiwania informacji w bazie danych utworzonej ze zbioru archiwów prawniczych z XIV w. W systemie istnieją dwie możliwości eksploatacji bazy danych:
— periodyczne wydawanie dokumentów w formie standardowej,



— udzielanie odpowiedzi na pytania użytkowników sformułowane w języku zbliżonym do naturalnego (tryb konwersacyjny).

Baza danych liczy 600 dokumentów. W systemie nie korzysta się z tekstów źródłowych (dokumenty są w oryginale napisane po łacinie lub w języku starofrancuskim) lecz ze streszczeń opracowanych przez specjalistów.

Odpowiedzi systemu udzielane są w czterech etapach:
— liczba dokumentów relewantnych do pytania
— kody dokumentów relewantnych
— streszczenia dokumentów wraz z komentarzami
— kompletny wydruk odpowiedzi.

Po każdym z tych etapów użytkownik może zmodyfikować pytanie dążąc do uzyskania zadowalającej odpowiedzi systemu.

Stosowany w systemie tezaurus liczy 1100 deskryptorów.

● **W Sécurité Sociale** (Ubezpieczalnia Społeczna) w Grenoble realizowany jest eksperyment polegający na łącznym przetwarzaniu danych, za pomocą samego systemu (SOCRATE), a pochodzących z trzech różnych instytucji: poborcy składek ubezpieczeniowych dostarczający informacji o prawach poszczególnych osób do świadczeń, kasy świadczeń pracowniczych oraz kasy świadczeń rodzinnych (dodatki rodzinne, pomoc socjalna). Każda z tych instytucji dysponuje własną bazą danych — prowadzone są zaawansowane prace mające na celu stworzenie wspólnej bazy danych. Dotychczas eksploatowana jest baza osób ubezpieczonych (330 000), baza osób pobierających świadczenia rodzinne (aktualnie 80 000 — liczba maksymalna 500 000) oraz baza lekarzy i personelu medycznego (aktualnie — liczba maksymalna 30 000). Na podstawie konsultacji tych baz dokonuje się rejestracji składek i wypłaty świadczeń. System działa, na maszynie IRIS 50 w trybie wsadowym, głównie z użyciem SOCRATE/COBOL.

* * *

SOCRATE jest systemem zarządzania bazami danych, oferującym użytkownikowi szereg udogodnień: język programowania zbliżony do naturalnego, magrogenerator, interfejs w języku wyższego poziomu, dwojaki tryb pracy (wsadowy lub konwersacyjny). Makrogenerator może w szczególności służyć do wprowadzenia wersji obcojęzycznej języka SOCRATE, np. polskiej. Struktura danych w systemie jest tego rodzaju, że pozwala uniknąć redundancji. SOCRATE może być implementowany na różnych maszynach; minimalna konfiguracja wynosi 98 Kb pamięci operacyjnej.

ZENON GRODZKI
MACIEJ WINIARSKI
Instytut Techniki Budowlanej

681.322.06:681.327.2:651.8

SYSFAP
Zintegrowany system automatycznie programowanych zbiorów danych

Artykuł opisuje, opracowany w Centrum Wdrażania ETO w Przemyśle Budowlanym w Liegé, system SYSFAP. System ten jest szczególnie przydatny do rozwiązywania złożonych problemów i nie wymaga wielkiego doświadczenia w programowaniu.

W Belgii powstał system SYSFAP zorientowany na dane, szczególnie przydatny do projektowania. Główną jego cechą jest to, że jest on niezależny od dziedziny projektowania i że może zostać wypełniony istniejącymi programami roboczymi, opracowanymi w różnych językach — poza systemem. Jest to możliwe dlatego, że punktami stycznymi programów roboczych są zbiory danych, których zarządzanie stanowi główne zadanie systemu.

Nazwa SYSFAP pochodzi od skrótu pełnej nazwy w języku francuskim: *Systeme Integre de Fichiers Auto-Programes*. Został on opracowany w latach 1967—71 pod kierownictwem p. G. Deprez w Centrum Wdrażania ETO w Przemyśle Budowlanym (Centre d'Edudes pour la Promotion des Ordinateurs dans la Construction, skrót — CEPOC). Centrum to jest częścią Uniwersytetu w Liege.

SYSFAP działa na komputerach IBM 360/175 i 360/185, a w przygotowaniu znajduje się jego adaptacja na komputery serii IBM 370. Wymagana pojemność pamięci operacyjnej wynosi 200 K bajtów. Dla rozbudowanych problemów pojemność szybkiej pamięci zewnętrznej mieszczącej BANK DANYCH powinna wynosić 50 Mln bajtów. Sam system SYSFAP mieści się na 2 wymiennych dyskach. Do eksploatacji ssytemu wymagane są ponadto 4 jednostki pamięci taśmowej.

Cel systemu

W przeciwieństwie do systemów ICES [1]) czy SAPRO [2]), celem SYSFAP jest nie tyle ułatwienie opracowania nowych programów przez użytkownika, co ułatwieniem użytkowania programów istniejących, opracowanych głównie poza systemem SYSFAP, w językach FORTRAN, PL-1, COBOL, ALGOL i innych, jakie dopuszcza system operacyjny komputera.

Poprzez możliwość automatycznej organizacji ciągów programów bez integracji programisty, system SYSFAP pozwala na rozwiązanie problemów kompleksowych, których nie rozwiązuje w całości żaden z dostępnych dla tego systemu programów indywidualnych.

Innymi słowy, celem SYSFAP'u jest udostępnienie potencjału obliczeniowego dużych komputerów inżynierom o słabym lub żadnym doświadczeniu w dziedzinie ETO.

Z tego też powodu SYSFAP nie ma języka problemowo zorientowanego w pełnym tego słowa znaczeniu ani kompilatora czy prekompilatora. Komunikaty (*messages*) o stosunkowo prostej syntaktyce, opracowywane przez specjalny dekoder, są wystarczającym środkiem do efektywnego wykorzystywania pełnych możliwości tego systemu.

Przy opracowaniu systemu autorzy brali pod uwagę fakt, że projektant nie jest zainteresowany samym programem, o ile tylko ma pewność, że jest on dobry.

Interesują go głównie dane i wyniki, a więc WEjście i WYjście. Podstawową ideą SYSFAP'u jest, aby z każdym zadaniem projektowym powiązać BANK DANYCH zawierający wszystkie informacje dostarczone o projekcie, lub uzyskane w trakcie obliczeń i stanowiące numeryczny model projektu, a następnie zorganizować cały system wokół tego BANKU. SYSFAP generuje więc i aktualizuje BANK DANYCH projektu.

W każdym zadaniu projektowym występują dane wejściowe, wyniki i — na ogół — szereg programów zorganizowanych w pętle, równoległe gałęzie i różne ich kombinacje.

Kluczem do zrozumienia SYSFAP'u jest to, że jest on zorientowany nie na programy lecz na dane (*data — oriented approach*). W SYSFAP'ie zadanie projektowe staje się uporządkowanym grafem zorientowanym. Zbiory są węzłami, a programy — gałęziami, służącymi do przekazywania danych z jednego zbioru do drugiego. Programy pełnią rolę „czarnych skrzynek", są anonimowe i mogą być wymienione na inne, bądź też zastąpione bezpośrednim wprowadzeniem zbioru danych w miejsce ich wyników. To właśnie, że programy komunikują się ze sobą jedynie za pomocą danych i wyników zamieszczonych w BANKU DANYCH, pozwala na praktyczną współpracę programów napisanych w różnych językach źródłowych.

Powiedzieliśmy, że SYSFAP jest zorganizowany wokół BANKU DANYCH (*Fichier General*). Zadaniem SYSFAP'u jest generowanie i aktualizacja tego BANKU. W tym celu system zawiera — niedostępny dla użytkownika — model logiczny projektu, który określa stosunki względem siebie różnych typów informacji zawartych w BANKU DANYCH. Nie tylko system, lecz także szczątkowy język problemowy jest zorientowany na dane, to znaczy składa się z rozkazów i pytań pod adresem zbiorów danych (np. żądanie wydruków, przesłania itp.).

[1]) ICES — Integrated Civil Engineering System, opracowany w latach 1964—67 w M.I.T. (Massachussetts Institute of Technology) pod kierownictwem prof. D. Roos'a.

[2]) SAPRO — System Automatyzacji Projektowania, opracowywany pod patronatem Ministerstwa Budownictwa CSRS w latach 1968—72, przy współpracy państw R.W.P.G., przez zespół pod kierownictwem J. Zavrela.

Mgr ZENON GRODZKI ukończył (1970) studia w zakresie matematyki na Wydziale Matematyki i Mechaniki Uniwersytetu Warszawskiego i w tym roku podjął pracę w Zakładzie Elektronicznej Techniki Obliczeniowej Instytutu Techniki Budowlanej. Od 1972 roku pełni funkcję kierownika Pracowni Metod zajmując stanowisko starszego asystenta. Specjalizuje się w realizowaniu na EMC zagadnień o charakterze kompleksowym. W r. 1974 jako członek zespołu autorskiego otrzymał nagrodę III Stopnia Ministra Budownictwa i Przemysłu Materiałów Budowlanych za opracowanie systemu optymalizacji konstrukcji stalowych hal przemysłowych — OSY.

Mgr inż. MACIEJ WINIARSKI uzyskał (1952 r.) dyplom na Wydziale Inżynierii Uniwersytetu w Cambridge. Po kilkuletniej praktyce projektowej w W. Brytanii, gdzie pełnił funkcje od asystenta do kierownika pracowni i realizował szereg projektów, w latach 1963—69 pracował w Biurze Studiów i Projektów Typowych Budownictwa Przemysłowego BISTYP a od 1969 r. — w Instytucie Techniki Budowlanej, gdzie obecnie prowadzi Pracownię Zastosowań w Zakładzie Elektronicznej Techniki Obliczeniowej. Autor szeregu nagrodzonych lub wyróżnionych projektów i realizacji.

Struktura SYSFAP'u

System składa się z: PROGRAMU NADZORUJĄCEGO (*Supervisor*), który wykonuje podwójne zadania:
— zarządza systemem i na sygnały programów sterujących ładuje żądany program do pamięci operacyjnej
— działa jako interfejs między SYSFAP'em a systemem operacyjnym komputera.

Ponadto, zawiera on PROCEDURY spełniające te funkcje, **które nie są wykonywane przez system operacyjny komputera,** PROGRAMY STERUJĄCE o jednakowym poziomie hierarchii, PROCEDURY ZARZĄDZAJĄCE BANKIEM DANYCH.

BANK DANYCH, zawiera w danym momencie wszystkie zaktualizowane dane o zadaniu projektowym (a więc wszystkie węzły grafu przedstawiającego to zadanie).

System dysponuje też zbiorem DANYCH SYSTEMOWYCH, odnoszących się wyłącznie do systemu oraz BIBLIOTEKĄ PROGRAMÓW, przechowującą programy w postaci modułów ładowania. SYSFAP posiada 3 poziomy: SUPERVISOR (poziom 1), programy sterujące (poziom 2), elementy robocze (poziom 3). Poziom programu jest każdorazowo sygnalizowany do SUPERVISOR'a.

Charakterystyka części składowych systemu SYSFAP

SUPERVISOR, jak wspomniano stanowi interfejs pomiędzy systemem operacyjnym komputera a systemem SYSFAP. Znajduje sie on stale w pamięci operacyjnej w czasie działania SYSFAP'u i wybiera programy, którymi steruje oraz zarządza.

Programy sterujące

W systemie SYSFAP istnieje 6 programów sterujących: WEJŚCIE, który wczytuje dane do BANKU, DEKODER, który przekodowuje komunikaty wejściowe na wymaganą formę wewnętrzną. SYMULATOR, który ustala rodzaj obliczeń jakie trzeba wykonać oraz ich kolejność RACHUNKOWOŚĆ, który rejestruje i prowadzi ewidencję czasu pracy systemu. WYJŚCIE — który umożliwia wybór odpowiednich jednostek urządzeń zewnetrznych komputera, OBLICZANIE — który steruje podprogramami roboczymi realizującymi właściwe obliczenia. Funkcja tych podprogramów jest niezależna od SYSFAP'u, znaczy to, że ich algorytmy obliczeniowe nie muszą być dostosowywane do SYSFAP'u. Każdy z programów sterujących ma bezpośredni dostęp do BANKU DANYCH.

BANK DANYCH wraz z procedurami zarzadzania tworzą jedna całość. Oprócz zbiorów przetwarzanych zawiera on tablice zawartości, kolejności uruchomień tak aby móc śledzić kolejność rozwiązywania projektu.

BANK DANYCH ma trzy stany:
— początkowy (przed rozpoczęciem obliczeń)
— aktywny (w czasie obliczeń)
— końcowy (po wykonaniu obliczeń).

Po zakończeniu cyklu obliczeń, stan początkowy zmienia sie w stan końcowy, a stan aktywny — w stan początkowy. Stan początkowy i końcowy przechowywane są na taśmach magnetycznych, natomiast stan aktywny — na dyskach.

Struktura danych i język SYSFAP'u

W każdym zadaniu projektowym dane są zgrupowane w zbiorach nazywanych ARTICLES, a które beda nazywane w tym artykule „zbiorami". Zbiory te są kilku typów i mają nastepujące nazwy:
— dane (*donnèe*) — zbiór nowych danych
— modyfikacja (*modification*) — zbiór zastępuje stare dane
— usuniecie (*supression*) — usuniecie zbioru istniejącego i wszystkich dalszych wyników na nim opartych
— wykonanie (*calcul*) — wykonanie obliczeń i wydawnictwo wyników określonych przez „rodzaj" i „przedmiot" (patrz niżej)
— wydawnictwo graficzne rysunków (*dessin*)
— wybór (*selection*) — wyniki zbiorów danych wyselekcjonowanych z uprzednio przetworzonych zbiorów.

Każdy zbiór (*article*) jest określony przez rodzaj (*nature*) i przedmiot (*sujet*). Rodzaj określony jest z kolei przez: rodzine (*famille*) i charakterystyke.

Przedmiot jest wyrażony poprzez nazwe. Podanie tych danych określa jednoznacznie numer zbioru.

Przykład: rodzina = „konstrukcje" } rodzaj
charakterystyka = (schemat obciążeń) }
przedmiot (nazwa) = „Rama 1 — 2"

Tak więc każdy zbiór w BANKU DANYCH może być zidentyfikowany przez podanie rodzaju i przedmiotu.

SYSFAP rozróżnia 5 rodzajów danych:
— koniecznych (*indispensables*)
— łączonych (*combinables*)
— równoległych (*synonimes*) — podaje się jedną z możliwości
— wybieranych (*optionnels*) — w przypadku gdy nie podano tych danych, są one zastępowane przez dane standardowe
— dowolnych (*facultatifs*) — nie branych pod uwagę przez system.

Język SYSFAP'u składa się z:
— separatorów (znak dziesiętny, znak równości, przecinek, spacja, które mają jednakowe znaczenie) np. 3.14 3,14 3=14 i 3 14 są równoważne
— słów (które znajdują się między dwoma separatorami, muszą się zaczynać od litery i posiadają znaczenie ustalone przez SYSFAP)
— nazw (jak wyżej, lecz których znaczenie ustala użytkownik)
— wartości (liczby całkowite lub rzeczywiste).

Liczby są zawsze zawarte między słowami, np.:
R = 300 m
KAT = 0.35 rad

Słowa pisane małymi bądź dużymi literami są drukowane na wyjściu dużymi. Istnieją jednak dwa wyjątki: pisane małymi literami słowa num i valeur zostają w wydruku zastąpione liczbami, np:
ELEMENT num PARAMETRE valeur
ELEMENT num LONGEUR valeur
zostaje zastąpione przez
ELEMENT 1 PARAMETRE = 95
ELEMENT 2 LONGEUR = 8.75 m

Jak widać z powyższego elementy języka są ograniczone. Z elementów tych wolno budować większe jednostki zwane komunikatami (*message*) według jednego tylko schematu. Komunikaty są bowiem zewnętrzną formą zbiorów, a ich nagłówek — adresem zbioru w BANKU DANYCH.

Komunikat jest ujęty w podwójne znaki i zawiera: nagłówek, komentarz, treść.

Struktura komunikatów jest jednakowa dla wszystkich elementów SYSFAP'u. Stanowią one jedyny sposób porozumiewania się z systemem.

Nagłówek zawiera
— typ komunikatu (jeden z 6 typów zbiorów danych wymienionych na początku niniejszego rozdziału)
— przedmiot (*sujet*)
oraz słowa łączące.

Komunikat pokrywa się ze strukturą zbiorów danych.

Komunikacja z systemem polega na wprowadzeniu zmian w zbiorach danych.

Nagłówek komunikatu identyfikuje odpowiedni zbiór w BANKU DANYCH.

Komentaż można opuścić. Zawiera on to, co użytkownik życzy sobie, aby zostało wydrukowane w tabulogramie wyników i musi być ujety w cudzysłów („ ").

Treść zawiera dane. Treść musza mieć komunikaty typu dane i modyfikacje, a mogą mieć komunikaty typu wybór. Komunikaty typu wykonanie, wydawnictwo graficzne i usunięcie nie zawierają żadnej treści.

Dwa przykłady nagłówka:	(1)	(2)
— typ	dane	dane
— rodzina	rama	oś

Charakterystyka — tablicowanie
przedmiot — (rama) Nr 4 (oś) N-S

Komunikat może być ogólny (dla całego rodzaju lub zbioru przedmiotów) lub szczególny zależny od słów łączących, którymi są: dla wszystkich (*pour tous*), dla każdego (*pour chague*)

Przykład komunikatu ogólnego:
typ (dla wszystkich) rodzina dla wszystkich
charakterystyka komentarz treść

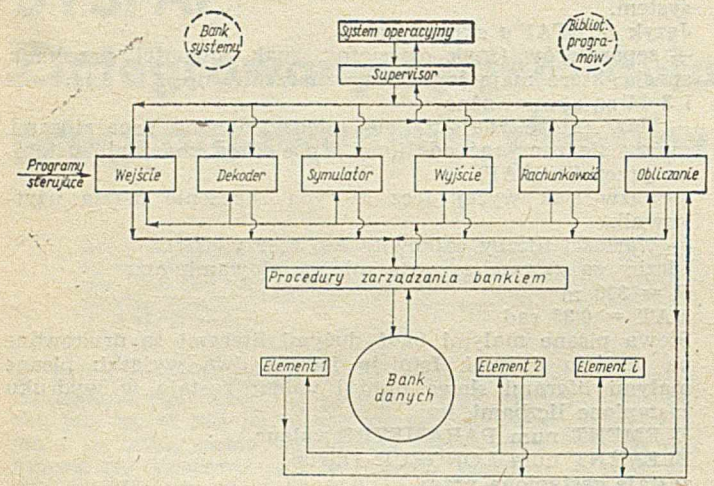
Przykład komunikatu szczegółowego:
typ rodzina charakterystyka przedmiot komentarz treść

Jak widać z powyższego w komunikatach ogólnych opuszcza się przedmiot.

Jak pracuje SYSFAP?

Przy uruchomieniu SYSFAPu program SUPERVISOR przegląda macierz kolejności uruchomienia programów, a ponieważ jest ona pusta w chwili uruchomienia, oddaje zarządzenie programowi sterującemu wejściem.

Program sterujący wejściem odczytuje zbiory *(articles)* oraz komunikaty wejściowe i zakłada BANK DANYCH rozpatrywanego zadania projektowego. Wszystkie komunikaty zostają początkowo zapisane w BANKU w formie zewnętrznej. Program sterujący wejściem oddaje zarządzanie SUPERVISOR'owi, wpisując do macierzy kolejności uruchomienia programu DEKODER (o ile wczytano komunikaty) lub w przeciwnym przypadku programu SYMULATOR.



W przypadku pierwszym uruchomiony przez SUPERVISOR zostaje program DEKODER. Wybiera on z BANKU te zbiory, które nie zostały jeszcze przetłumaczone, identyfikuje je, wybiera odpowiednią tablicę syntaktyczną i zapisuje te zbiory ponownie w BANKU DANYCH — tym razem jednak w odpowiedniej postaci wewnętrznej, usuwając jednocześnie zbiór w postaci zewnętrznej. W przypadku komunikatu typu usunięcie lub modyfikacja DEKODER przegląda BANK DANYCH i usuwa z niego odpowiednie zbiory. W końcu, DEKODER wpisuje nazwę programu SYMULATOR do tablicy uruchomień i zwraca zarządzanie SUPERVISOR'owi.

Program SYMULATOR ma kluczowe znaczenie w pracy systemu SYSFAP.
Zadanie SYMULATORA jest dwojakie:
— ustalić listę i kolejność programów, jakie należy wykonać tak, aby zrealizować zadanie projektowe użytkownika.
— ustalić listę brakujących danych.
Do tego celu SYMULATOR dysponuje trzema tablicami:
— tablicą zawartości BANKU DANYCH (stan aktualny)
— tablicą, która przedstawia listę potrzebnych użytkownikowi zbiorów (końcowych i pośrednich)
— tablicą z wykazem programów bibliotecznych włączonych do systemu SYSFAP wraz z listą ich danych początkowych i końcowych.

Dla przykładu, przedstawiamy zadanie projektowe zakładając, że użytkownik potrzebuje zbioru wyników Nr 400, który jest wytwarzany przez program (element) F, a który z kolei wymaga jako danych zbiorów 80, 300 i 320. Sprawdzając tablicę zawartości BANKU DANYCH program SYMULATOR sprawdza czy zbiory te w nim się znajdują, SYMULATOR po wpisaniu do tablicy uruchomień programu F zwraca zarządzanie SUPERVISOR-owi.
Jeśli brakuje któregoś z wyżej wymienionych zbiorów SYMULATOR sprawdza, czy istnieją zbiory poprzedzające poszukiwany zbiór i zbiory dalsze, aż do początku drzewa grafu. I tak np. zbiór 300 może być osiągnięty albo programem E albo J, a śledząc dalej poprzez ścieżki A-B-C-D-E-F lub A-E-C-G-H-I-J-F SYMULATOR wytwarza zwykle jedną ścieżkę, z programów dostarczonych przez użytkownika. Jedynie w ten sposób użytkownik ma wpływ na programy, których chce użyć i zbiory, które chce otrzymać. Jeśli brakuje któregoś ze zbiorów, SYMULATOR wydrukuje brakującą gałąź grafu (tj. listę brakujących programów) i użytkownik może je włączyć do systemu przy następnym przeliczeniu. Gdy wszystkie potrzebne zbiory są już w BANKU, SYMULATOR wpisuje je do tablicy kolejności uruchomień (aktywizacji) i oddaje zarządzanie SUPERVISOR'owi, który przekazuje je z kolei programowi MASTER CALCULATOR sterującemu obliczeniami i zarządzającemu ich wykonaniem przez programy użytkowe włączone do SYSFAP'u. Po przeliczeniu wszystkich programów wybranych przez SYMULATOR, program sterowania obliczeniami oddaje w podobny sposób zarządzenie SUPERVISOR'owi, a ten uruchamia kolejno program RACHUNKOWOŚĆ. Program WYJŚCIE uwzględnia wszystkie życzenia użytkownika dotyczące wyboru i sposobu prezentacji wyników; wybiera je z BANKU DANYCH i drukuje przy uwzględnieniu wskazanych przez użytkownika jednostek. Program RACHUNKOWOŚĆ odnotowuje czas, przez który użytkownik używał SYSFAP'u oraz na żądanie wystawia fakturę za korzystanie z systemu. Po wykonaniu wszystkich życzeń użytkownika, aktualny BANK DANYCH zostaje przeniesiony z dysków na taśmę magnetyczną i obliczenia kończą się.

Włączenie nowych programów do SYSFAP'u

Włączenie dodatkowych programów użytkowych do SYSFAP'u dokonuje programista. Następuje to przez dodanie dwóch wstawek, stanowiących interfejs na wejściu i na wyjściu. Wstawki te umiejscawiają potrzebne dla programu zbiory w BANKU DANYCH i przenoszą je w odpowiedniej kolejności do zbiorów roboczych programu oraz odsyłają wyniki do BANKU. W ten sposób nic nie zmienia się wewnątrz włączonego programu roboczego, z wyjątkiem np. jednostek, które muszą być zgodne z jednostkami używanymi w systemie. Instrukcje zostają pozbawione formatów a sygnalizacja charakteru błędów musi być skierowana do specjalnej procedury o nazwie EXAT.
Rozróżnia się dwa typy błędów:

1) lokalne — na poziomie elementu i niezależne od SYSFAP'u; są one sygnalizowane przez program użytkowy,

8) podstawowe — element odsyła błąd przez procedurę EXAT do programu sterującego, a ten — do SUPERVISOR'a.

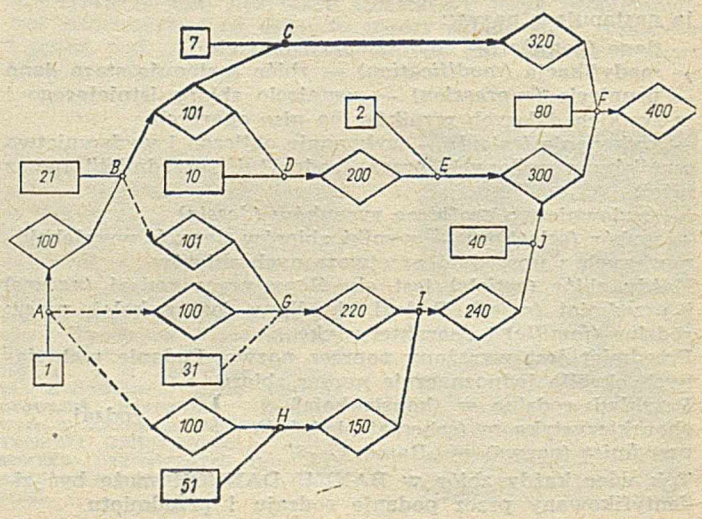
Wyjścia programu zostają zamienione na przesłanie do BANKU DANYCH. W ten sposób, program roboczy, stanowiący element systemu, ma 3 części — interfejs na wejściu — program właściwy — interfejs na wyjściu.

Program roboczy może użytkować całość pamięci wewnętrznej (bez obszarów zajętych przez System Operacyjny komputera oraz program SUPERVISOR) pod warunkiem opróżnienia jej w chwili oddania zarządzenia jednemu z programów sterujących. Wewnątrz samego systemu SYSFAP, włączenie nowego elementu wymaga uzupełnienia tablicy uruchomień. Zmiany te wykonują specjalne programy poprawkowe.

Program sterujący obliczeniami odnajduje interfejs programu na wejściu, program właściwy i interfejs na wyjściu oraz uruchamia je w tej kolejności. Po wykonaniu obliczeń program sterujący obliczeniami uzupełnia tablicę zawartości BANKU DANYCH nazwami nowo dodanych zbiorów.

Podsumowanie

Korzyści jakie daje SYSFAP użytkownikowi to:
— automatyczny wybór programów, ich kolejności i przeprowadzenie obliczeń z możliwością ingerencji z zewnątrz. Od użytkownika jest wymagane tylko dostarczenie danych wejściowych i specyfikacji żądanych wyników i to w bardzo prostym języku



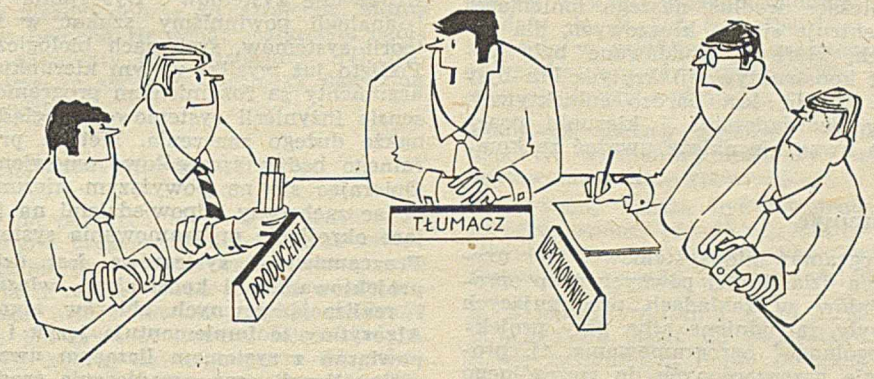
— możliwość modyfikacji projektu przy minimum podawania nowych danych oraz automatyczne uaktualnienie danych modyfikowanych bezpośrednio lub pośrednio
— gospodarka BANKIEM DANYCH, która jest zautomatyzowana tak dalece, że użytkownik może nie zdawać sobie sprawy z jego zawartości
— możliwość wyboru wydawnictwa wyników.

Warto podkreślić jeszcze raz, że w przeciwieństwie do innych systemów zintegrowanych, SYSFAP nie ma problemowo zorientowanego języka programowania, ani związanych z nim kompilatora i prekompilatora, gdyż celem SYSFAP'u nie jest ułatwienie pisania, lecz użytkowania programów. Narzędziami roboczymi systemu są programy napisane w dowolnym języku (jednym z tych, którymi dysponuje system operacyjny komputera), zwykle już istniejące poza systemem SYSFAP. W ten sposób duża ilość programów może być włączona do SYSFAP'u bez zmian.
Możliwość wprowadzania programów w COBOL'u oznacza również, że system SYSFAP można stosować nie tylko w dziedzinie obliczeń inżynierskich i nie tylko do projektowania.

Jeśli idzie o dynamiczne zarządzanie pamięcią, to nie ma go na poziomie systemu SYSFAP i nie przewiduje się jego opracowania. Problem ten zostanie rozwiązany automatycznie, gdy wejdą w życie komputery z pamięcią wirtualną. Natomiast na poziomie elementu, dynamiczne zarządzanie pamięcią nie jest ograniczone, gdyż podczas pracy podprogramu roboczego system SYSFAP jest wyłączony i jeśli element używał pamięci w sposób dynamiczny poza systemem, to będzie ją używać bez przeszkód i wewnątrz systemu.

Bardziej szczegółowy opis systemu wymagałby 3—4 artykułów tego samego rozmiaru i byłby zbyt obszerny dla szerszego grona Czytelników INFORMATYKI. Mimo ogólnego przeznaczenia systemu, autorzy SYSFAP'u zastosowali go w pierwszym rzędzie do projektowania dróg; przytoczone przykłady z tej dziedziny odbiegałyby poziomem szczegółowości od reszty artykułu i mogłyby interesować tylko wąski krąg fachowców. Dlatego przedstawiono pracę systemu na przykładzie ogólnym, niezależnym od dziedziny zastosowania.
Wyrażamy podziękowanie Dr G. Deprez za zgodę na niniejszą publikację i udzielone informacje, oraz doc. dr hab. inż. Z. Leśniakowi za zainteresowanie nas tym systemem oraz nadesłane materiały.
SYSFAP nie był dotychczas stosowany w Polsce, ale jego ciekawa koncepcja zosługuje na uwagę.
Czytelnicy mający styczność z opracowywaniem systemów przeznaczonych do rozwiązywania problemów o złożonym charakterze, wiedzą jak pracochłonne i żmudne jest to zadanie; najczęściej włączenie istniejącego w danej dziedzinie oprogramowania jest bardzo trudne i łatwiej jest w takich przypadkach ułożyć nowy program niż dostosować do własnych potrzeb program o innej strukturze, choć o podobnym przeznaczeniu. Możliwość tworzenia na bazie istniejącego (napisanych w różnych językach źródłowych) oprogramowania rozwiązującego kompleksowo dane problemy jest główną zaletą SYSFAP'u i tę możliwość należy brać pod uwagę w pracach nad systemami problemowymi, „zorientowanymi na dane".
Wspólne cechy z SYSFAP'em posiadają mniejsze systemy, które opracowują autorzy artykułu, co im ułatwiło zrozumienie intencji autorów SYSFAP'u.

WIADOMOŚCI PKAPI



Prezydium PKAPI w Rzeszowie

Marcowe (3.III.75 r.) Prezydium Polskiego Komitetu Atomatycznego Przetwarzania Informacji pod przewodnictwem prof. dr inż. Andrzeja Straszaka poświęcone było w pierwszym rzędzie rozwojowi informatyki na rzeszowszczyźnie. Temat ten podyktowany jest nie tylko tradycją szerszego omawiania spraw regionu w miejscu spotkań prezydialnych PKAPI. W tym roku zaczyna się podsumowywanie realizacji Programu Rozwoju Informatyki na lata 1971—75.
W krótkim wprowadzeniu do narady przewodniczącego Oddziału Wojewódzkiego PKAPI w Rzeszowie dr inż. J. Woźniackiego sprawa nie przedstawia się różowo. Tylko powiaty północne rzeszowszczyzny — Mielec, Stalowa Wola, Tarnobrzeg oraz na południu — Krosno, Jasło i Rzeszów korzystają z dobrodziejstw komputeryzacji. Jednak zbyt dużo jeszcze „białych plam" na informatycznej mapie regionu, by można było mówić o pozytywnych wynikach realizacji Programu.
Wśród koncepcji informatycznego zagospodarowania rzeszowszczyzny najciekawsza jest propozycja zainstalowania komputerów w bardziej rozwiniętych powiatach, które mogłyby „obsługiwać" również powiaty sąsiadujące. Zdaniem prof. A .Straszaka, rozwiązanie to dałoby rzeszowszczyźnie rolę wiodącą w rozwoju informatyki przemysłowej w takiej formie.
Z kolei ZETO — Rzeszów zorganizowało z dobrym skutkiem swój oddział w Krośnie, a w zamiarze ma powołanie filii w Dębicy i Przemyślu. Jest to zresztą sprawdzona droga do informatyzacji. Kroczą nią inne województwa, zwłaszcza ZETO-Wrocław mające już kilka filii w miastach powiatowych. Wymaga to jednak nie tylko dobrej woli, ale i poważnych tytułów inwestycyjnych. Tymczasem samo ZETO — Rzeszów funkcjonuje w charakterze niewygodnego sublokatora spontanicznie rozrastającej się Politechniki Rzeszowskiej...
W dalszej części zebrania uczestnicy omówili bieżące sprawy krajowe. Przeważały głosy na temat trudności w uruchamianiu maszyn serii ODRA, najczęściej spowodowane nieterminowymi przyjazdami ekip uruchamiających. Przekazano też informacje o przygotowaniach do III krajowej Konferencji Informatyków i VII Kongresu Techników Polskich. Między innymi na III KKI przedstawione będą doświadczenia informatyczne Wytwórni Sprzętu Komunikacyjnego — Rzeszów. PKAPI Kraków czyni starania o przydział minikomputerów MERA 303 i 304 do obsługi VII Kongresu Techników Polskich, a w sali obrad VII Zespołu Kongresu czynna będzie wystawa sprzętu informatycznego.

Po zakończeniu obrad uczestnicy Prezydium zwiedzili Ośrodek ETO w WSK i ZETO w Rzeszowie.

Termin zwołania następnego Prezydium ustalono na 17 kwietnia w Krakowie. (żel)

Programowanie systemowe — nowopowstająca dziedzina nauki*)

Programowanie systemowe staje się obecnie kluczową gałęzią informatyki. W artykule przedstawiono te dziedziny programowania systemowego, które rokują nadzieję, że wkrótce staną się dyscyplinami naukowymi. Zawarto również uwagi na temat metod nauczania programowania systemowego.

We wszystkich dziedzinach ludzkiego działania niezadowolenie było zawsze główną przyczyną i podnietą do lepszego zrozumienia i szukania dróg poprawy. W ostatnim dziesięcioleciu rosnąca wciąż liczba informatyków odczuwała potrzebę lepszych metod pracy. Powszechnie znane były niepowodzenia nawet nie najbardziej ambitnych projektów oprogramowania. W związku z tym rodziły się wątpliwości co do solidności zawodu informatyka. W szczególności stawiano pytanie: czy ludzie zajmujący się projektowaniem i produkcją programów systemowych mają wystarczające doświadczenie i odpowiednie narzędzia, ażeby mogły powstać dobre produkty?

Programistów systemowych i ich kierowników pytano, czy mają środki i wypracowane metody postępowania; mówiono im, że — aby utrzymać swą pozycję — muszą przyswoić sobie całkowicie nowy styl myślenia w pracy.
Czy wpływ długo odczuwanego niezadowolenia z często złych wyników prac w dziedzinie oprogramowania jest na tyle silny, aby możliwe było rozwinięcie i wprowadzenie, często omawianych teraz, nowych skutecznych zasad i metod koniecznych do zmiany sytuacji?

Spróbuję zdefiniować i opisać dziedzinię programowania systemowego. Nie widzę jednakże pilnej potrzeby szkolnego wykładu obejmującego w równym zakresie wszystkie rodzaje działalności, należące według naszego mniemania do tej dziedziny. Skoncentruję się na kluczowych, dla całej dyscypliny, gałęziach, które to poddawane były poprzednio ostrej (czasami konstruktywnej) krytyce lub były całkowicie lekceważone. Wybór jest bardzo subiektywny. Miały na niego wpływ doświadczenia i kierunki pracy Autora. Przedstawionych uwag nie należy uważać za kompletne.

Programowanie systemowe

Początkowo termin „programowanie systemowe" był używany głównie jako nazwa działalności pewnych grup oprogramowania lub wydziałów w zakładach produkujących komputery. Grupy te były zatrudnione albo przy projektowaniu i produkcji „ogólnego' oprogramowania, tj. programów lub pakietów nie przeznaczonych do szczególnego zastosowania, albo przy większych projektach sprawiających wielkie trudności „organizacyjne". Tak więc programowanie systemowe było rozumiane w tych czasach jako pisanie kompilatorów, programów usługowych, bibliotek programów WE/WY, programów ładujących i systemów „monitorujących".

Bardzo często „programowanie systemowe" było nazwą specjalizacji na wydziałach informatyki *(computer science)*. Już od dawna podejmowano próby [1] identyfikowania i uczenia zasad projektowania i konstruowania, zamiast prezentowania w specjalistycznym żargonie suchych faktów i oderwanych metod.

Na uwagę zasługuje raport „Program nauczania 68" [2], opracowany przez Komitet ds. Programu Nauczania Informatyki ACM. Przedstawiono w nim projekty programów nauczania różnych gałęzi informatyki. W raporcie termin „programowanie systemowe" był jednakże używany tylko w wąskim zakresie. Punkt zatytułowany „Programowanie systemowe" (w raporcie o numerze 14) ogranicza się do podania zwykłych trybów użytkowania komputerów uniwersalnych oraz przeglądu wykrytych zjawisk i metod stosowanych w dużych systemach operacyjnych.
Z opisanego powyżej programu kursu możnaby wnioskować, że odpowiedniejszą nazwą byłoby „Pojęcia, Możliwości i Metody Implementacyjne w Systemach Operacyjnych" wraz z uwagą, że nie jest uwzględniony szczegółowy opis programów przetwarzających, np. kompilatorów.
Wydaje się, że w punkcie 14 położono nacisk na omówienie metod a za mało uwagi poświęcono na problemy projektowania i konstruowania systemów w znaczeniu inżynierskim.
W 1972 r. ukazała się książka J. J. Donovana „Programowanie Systemowe" [3]. Wyrosła ona z długiej tradycji kursów w MIT [1]. Warto nadmienić, że obejmuje ona szerszy zakres zagadnień. Zawiera zasadnicze części sześciu punktów wspomnianego „Programu 68", a mianowicie „Komputery i programowanie" (B2), „Języki programowania" (12), „Programowanie systemowe" (14), „Konstrukcja kompilatorów" (15), „Wstęp do organizacji" (A8).
W tym czasie robiłem korektę swojej książki obejmującej wstępne informacje z dziedziny programowania systemowego (4) i bardzo byłem zadowolony, widząc uderzające podobieństwo w wyborze tematów w obu książkach. Daje się zauważyć, że termin „programowanie systemowe" może być rozumiany szerzej niż dotychczas.

Każdy, kto zajmuje się nauczaniem programowania systemowego prawie natychmiast dochodzi do wniosku, że samo prezentowanie recept i metod jest bezużyteczne i całkowicie nieuzasadnione. Z punktu widzenia technicznego i psychologicznego trzeba ten materiał ukazać w szerszym kontekście.

Metody programowania są zawsze stosowane w procesie konstruowania jakiegoś „systemu" lub „podsystemu". Dlatego naturalne jest poszukiwanie właściwego kontekstu w dziedzinie, którą moglibyśmy nazwać „projektowanie i konstruowanie systemów". Dyscyplina taka dopiero powstaje i analogii powinniśmy szukać w inżynierii systemowej, teorii systemów, systemach biologicznych i innych.
Podjęto już wysiłki w tym kierunku: np. w [5] podane są argumenty za rozumieniem programowania systemowego w sensie inżynierii systemowej. Poglądy te nie zyskały jednakże dużego znaczenia. Metoda programowania strukturalnego będzie szczegółowo omówiona poniżej.
Opierając się na powyższym kierunku rozumowania oraz będąc zachęcony odpowiedziami na [6] proponuję następujące określenie programowania systemowego:

Programowanie systemowe jest dziedziną zajmującą się projektowaniem i konstrukcją, własnościami, reprezentacją i realizacją pewnych zbiorów algorytmów systemowych. Algorytmy te implementują język i konwencje operacyjne powiązań z systemem liczącym uwzględniając konieczność optymalizacji oraz ograniczenia środowiska.

W określeniu tym nie występuje termin „metody". Dziedzina projektowania i konstruowania algorytmów systemowych występuje na tym samym poziomie, co reprezentacja [2], własności [3] i realizacja [4] tych algorytmów. Być może dziedzinę projektowania i konstruowania (obejmującą także problemy zarządzania związane z budową systemów) powinno się wyeksponować jeszcze bardziej.
Programowanie systemowe obejmuje więc systemy operacyjne, kompilatory itp.; są one rozpatrywane jako części logicznej całości, która być może jest częścią jeszcze większej całości. Wynika stąd, że nacisk jest położony na stronę systemową. Rozwiązania muszą być z konieczności zapożyczone z innych dziedzin, na przykład języków formalnych, psychologii językowej, teorii informacji, inżynierii systemowej, teorii projektowania, teorii algorytmów, teorii grafów, teorii prawdopodobieństwa, analizy numerycznej itp., a także z dziedziny projektowania urządzeń elektronicznych.
W dalszej części artykułu podkreślone będą aspekty projektowania i konstruowania w programowaniu systemowym.

*) opracowano na podstawie artykułu G. Seogmüllerapt. „Systems Programming as an Emetging Discipline", opublikowanego w IFIP Information Processing 74 — software.

1) Massachussetts Institute of Technology.

2) tzn. wszystkie problemy językowe

3) włączając potrzeby przestrzenno-czasowe oraz działanie dynamiczne

4) tzn. środki i sposoby, dzięki którym wykonywane są algorytmy

Cechy charakterystyczne omawianej dziedziny

Dla nowej dziedziny poznania naturalne jest, że po powstaniu pierwszych prac następuje okres entuzjastycznych badań i odkryć (nazwiemy go okresem 1), w którym każdy chce zbadać możliwości nowych urządzeń, nie zwracając uwagi na takie rzeczy, jak dyscyplina, bezpieczeństwo, problemy związane ze wzrostem wielkości, właściwe miejsce w większym systemie. Ponieważ łatwo uzyskać sukcesy pojawiają się (okres 2) ludzie, którzy naiwnie uogólniają uzyskane dotychczas wyniki, produkują „intuicje", których zastosowanie, często „całkiem niespodziewanie" nastręcza mnóstwo problemów. W międzyczasie pewni ludzie nie związani bezpośrednio z programowaniem systemowym wyrabiają sobie o nim niezależny sąd. Narasta konflikt. Sytuacja się zmienia, wielu bardziej lub mniej kompetentnych ludzi ogłasza „kryzys" (okres 3). Większość z nich jest po cichu bardzo zadowolona z tego stanu rzeczy, który „przewidzieli już dawno temu". Okres drugi i trzeci są to czasy fałszywych proroków, łatwych pieniędzy i szarlatanów. Zwykle dopiero po pewnym okresie dostrzega się tych nielicznych (okres 4), którzy głoszą skromność, pokorę [36], zwracają uwagę na konsekwencje, którzy udzielają rady, w jaki sposób „mądrze używać" „silnych" narzędzi, którzy ukazują niebezpieczeństwa nieograniczonego używania urządzeń oraz propagują przemyślane ograniczenia.

Okres czwarty jest początkiem dojrzałości. Dziedzina zaczyna być rozwijana. Na używane poprzednio bez ograniczeń i zastanowienia narzędzia nakłada się pewne naukowo uzasadnione ograniczenia. W ten sposób konsekwencje użycia tych narzędzi są lepiej widoczne i możliwe do przewidzenia. Praca w ramach dziedziny nauki oznacza, że o każdej decyzji można powiedzieć, dlaczego została podjęta w ten, a nie w inny sposób. Intuicyjne i nieuzasadnione przeczucia, co jest dobre, a co złe (w okresie od 1 do 3) zastąpiono przez ścisłe rozumowanie.

Wprowadzenie schematów rozumowania powoduje powstanie struktury w dziedzinie, która jej nie posiada albo posiada strukturę jeszcze niezbyt dobrze wykształconą. Można tutaj zastosować uwagę Brincha Hensena: „całkowita ogólność pociąga za sobą całkowity brak struktury" [7]. Należy zdawać sobie sprawę ze związku pomiędzy dziedziną nauki a istnieniem struktury.

Projektowanie i konstruowanie systemów

Ograniczymy się tutaj do pewnych uwag o projektowaniu i konstruowaniu systemów programowania, chociaż *mutatis mutandis* cały materiał jest równie ważny dla sprzętu i systemów liczących jako całości. Chcąc ustalić istotę pojęcia systemu nie mogą nas zadowolić zwykłe definicje encyklopedyczne, takie jak „system jest całością składającą się z części ułożonych według pewnego planu lub schematu" czy też „...zbiór lub uporządkowanie przedmiotów tak ułożonych lub powiązanych, że tworzą jedność lub organiczną całość", które są za ogólne i za bardzo statyczne. W książce Jenkinsa i Youle'a „Inżynieria programowania" [8], możemy znaleźć o wiele lepiej brzmiącą definicję systemów:

1. System jest złożonym ugrupowaniem zasobów w postaci ludzi, pieniędzy, maszyn i materiałów
2. Systemy można rozbijać na podsystemy: liczba podsystemów zależy przy tym od badanego zagadnienia
3. Praca danego podsystemu oddziaływuje na inne podsystemy i dlatego podsystem nie może być projektowany w izolacji od całego systemu
4. Badany system tworzy zwykle element hierarchii złożonej z podobnych systemów
5. System, aby funkcjonować musi posiadać ogólny cel, będący kompromisem pomiędzy sprzecznymi ze sobą celami. Na ogólny cel wpływa szerszy system zdeterminowany przez wyższe w hierarchii systemy
6. System musi być zaprojektowany tak, aby osiągał swój cel w możliwie najlepszy sposób i to pozwoli na działanie efektywne.

Na podstawie powyższych definicji można wymienić następujące główne własności systemu:

a) określenie go przez zwięzły zbiór niesprzecznych ze sobą ogólnych celów (powyższy punkt 5)

b) obecność struktury (punkty 1 i 2), która jest „funkcjonalna" (punkt 3), a nie „opisowa", tzn. nie zależy od notacji lub reprezentacji i która jest zwykle hierarchiczna (punkt 4)

c) istnieje potrzeba „optymalizacji" (punkt 6).

Wydaje się, że kryteria te stosują się równie dobrze do systemów skonstruowanych przez człowieka, jak i systemów, które wykryliśmy w przyrodzie (patrz 2).

W konsekwencji projektowanie i konstruowanie systemu zakłada określenie zwartego zbioru twierdzeń o własnościach sprecyzowanych w punktach a, b i c. Jak wiemy, proces ten ma zwykle charakter iteracyjny oraz jest długi i bolesny. Czasami wysiłki mogą okazać się bezużyteczne, jeżeli nie uzyska się zadowalającego zbioru twierdzeń.

Jakie zasady możemy stosować, aby ułatwić projektowanie i konstruowanie? Jest to pytanie zasadnicze. Można wymienić różne metody, np. użycie tablic decyzyjnych, analizy ścieżki krytycznej, symulacji, prognozowania itp. Jest to jednakże mylące i odwraca uwagę od głównego celu, ponieważ metody te zakładają istnienie „myśli przewodniej", która „wie" jak wykonać prawidłowo pracę. Cytuję powtórnie Jenkinsa i Youle'a: „Nie może być podejścia systemowego, jeżeli za duży nacisk kładzie się na metody, a nie na rozwiązanie problemu" [8]. Musimy więc szukać pomocy w dziedzinie samego rozwiązywanego problemu, w szczególności w dziedzinie projektowania i konstruowania systemów.

Według wiedzy autora nie istnieje wiele bezpośrednich pomocy. Jednakże w cytowanym drugim punkcie definicji systemów, który mówi: „System można rozbić na podsystemy; liczba podsystemów zależy przy tym od badanego zagadnienia" zawarta jest wartościowa wskazówka. Możliwy jest podział problemu na podproblemy. Cały problem może być rozwiązany przez kolejne rozwiązanie podproblemów nie wymagające rozważania wszystkich szczegółów całego systemu. Jest to stara zasada „dziel i rządź"; wprowadził ją do dziedziny programowania Dijkstra i teraz jest szeroko rozpowszechniona jako zasada „poziomów abstrakcji" [10, 11, 12, 13, 14]. Wokół tej idei powstała metoda projektowania i konstruowania systemów oprogramowania, zwana „programowaniem strukturalnym". Obecnie wydaje się, że jest to najbardziej obiecująca metoda.

Ostatnio w dziedzinie programowania strukturalnego można zaobserwować coraz większe łączenia zasad projektowania i konstruowania ze stroną językową.

Ponieważ chcemy możliwie dokładnie oddzielić te dwa aspekty, wymienimy najpierw najważniejsze zasady projektowania i konstruowania.

A. Systemy muszą być konstruowane i implementowane (programowane) przede wszystkim poprawnie. Nie jest możliwe otrzymanie prawidłowego systemu poprzez późniejsze testowanie (zasada „poprawności *a priori*").

B. Z punktu widzenia zasad strukturalności i podzielności powinno się dążyć do podsystemów, które posiadają możliwie najmniejszą liczbę połączeń i oddziaływań na inne podsystemy. Takie podsystemy mogą być najbardziej niezależnie opracowywane w szczegółach i sprawdzane.

C. W czasie procesu projektowania i konstruowania największą uwagę trzeba poświęcić na to, aby decyzje były zawsze podejmowane na właściwym poziomie. (W szczególności, jeśli odwołujemy się do obiektów systemu lub mają być podjęte decyzje dotyczące obiektów systemu, trzeba spróbować robić to możliwie niezależnie od reprezentacji i sposobu implementacji tych obiektów).

Punkty B i C są specjalnymi przypadkami zasady „minimum zobowiązań".

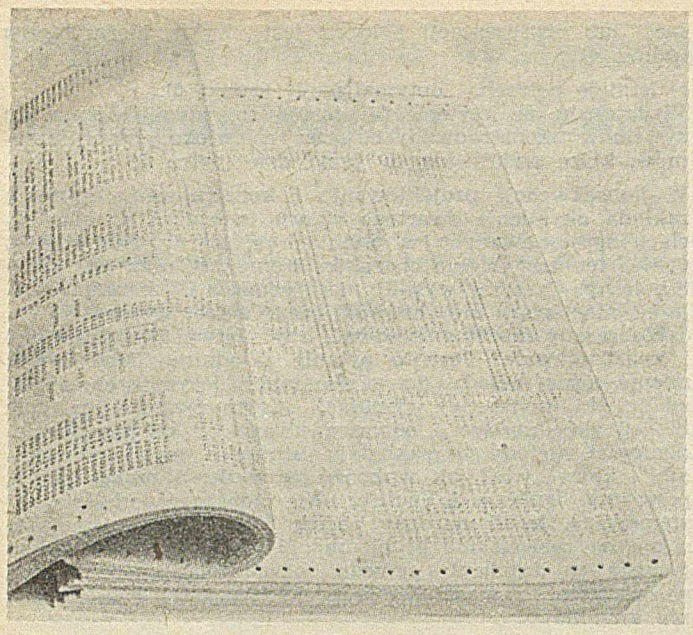
D. Dobry pomysł posiada pewien stopień ogólności, tzn. zastosowanie go do konstrukcji specyficznego systemu powoduje zastosowanie go w innych przypadkach.

E. Późniejsze adaptacje i modyfikacje podsystemu powinny być w możliwie największym stopniu przewidziane i ułatwione.

F. Płodnym sposobem rozumowania, pomocnym w wykazaniu poprawności uporządkowania operacji lub działań systemu jest wykorzystanie tych relacji pomiędzy obiektami systemu, które są stałe ze względu na pewne grupy operacji lub działań (np. [37], [38]).

Wyliczanie i indukcja mogą być również pomocą.

G. Poziomy abstrakcji stanowią „zamknięte obszary rozumienia" [14], tzn. na każdym poziomie znajduje się dobrze zdefiniowany zbiór operacji na dobrze zdefiniowanym zbiorze obiektów. Innym sposobem wyrażenia tego faktu jest wprowadzenie pojęcia „maszyny wirtualnej" związanej z danym poziomem. Pojęcie to może być pomocne w rozu-



mieniu, projektowaniu i konstruowaniu „styków" pomiędzy programowymi składnikami należącymi do różnych poziomów.

Jasne jest, że punkty te podkreślają znaczenie metody programowania strukturalnego w rozpracowywaniu sposobów rozumowania przy projektowaniu i konstruowaniu oprogramowania. Dzięki tej metodzie, wprowadzane są zasadnicze elementy, które sprawiają, że programowanie systemowe staje się dyscypliną naukową.

Nieuczciwością byłoby nie dodać tu kilku krytycznych uwag. Metoda programowania strukturalnego dostarcza tylko pośrednich zasad dla metodologii projektowania i konstruowania. Nawet w szerszej dziedzinie, inżynierii oprogramowania przejawia się to tym, że gdy ktoś chce być konkretny, pisze zdania, z którymi nie można się zgodzić, np. w [8]: „Nie ma tu miejsca na czyste myślenie, lecz...". Spotykając się z tak dużą ilością niejasności czujemy się trochę zagubieni. W szczególności nie możemy spodziewać się pomocy, gdy próbujemy zbudować zbiór celów niesprzecznych dla całego dużego i złożonego systemu, np. systemu operacyjnego. Aby otrzymać jasny, jednolity, funkcjonalnie akceptowalny projekt konstrukcyjny nie wystarczy wyeliminować tylko sprzeczne cele. Nie ma reguł na otrzymanie akceptowalnego i niesprzecznego zbioru celów. „Początkowe fazy projektowania" są wyrodnym dzieckiem programowania strukturalnego (i nie tylko). Stosunkowo słabo rozwinięte są sposoby reakcji na błędy, a także komunikacja pomiędzy oddzielnymi hierarchiami. Zmieniając trochę temat: czy poziomy w strkturze systemu operacyjnego wynikające z różnych strukturalnych kryteriów (np. brak blokad) stopniowy rozwój indywidualnych maszyn użytkowników korzystających z jednej maszyny rzeczywistej są zawsze zgodne?

Nie będziemy jednak wgłębiać się w ten problem.

W literaturze opisanych jest tylko kilka większych projektów, w których zastosowano dyscyplinę programowania strukturalnego. Prócz raportu Dijkstry o systemie THE [10], istnieje kilka raportów o IBMowskiej metodzie „Zespół Głównego Programisty" *(Chief Programmer Team)*, wymienionych w [28], istnieją raporty o próbach w Toronto z systemem SUE [29], a także dostępne są krótkie sprawozdania o projekcie BSM z Monachium, który zakłada zbudowanie bardzo dużego wielofunkcyjnego systemu operacyjnego [30].

Język i struktura systemu

Jak zauważyliśmy w punkcie b) poprzedniego rozdziału system posiada strukturę funkcjonalną. Nie wiadomo czy zawsze można znaleźć, używając elementów języka, taki izomorficzny opis, aby przeglądając statyczny tekst można było łatwo zrozumieć strukturę funkcjonalną. Korzyści płynące z uwidocznienia zdarzeń przez statyczny opis (lub za pomocą innych środków) są niewątpliwe. Jesteśmy jednakże daleko od stworzenia jakiegoś akceptowalnego schematu lub narzędzia nadającego się do tej pracy.

Musimy rozróżnić mechanizmy językowe pomocne w procesie tworzenia od tych, które są wykorzystywane w późniejszej konsekwencji systemu oraz przy wprowadzaniu zmian. Ostatnio ukazało się wiele artykułów z dziedziny tworzenia systemów (np. wybrane arbitralnie [12], [15]—[22], [35]). Nie można znaleźć jednakże materiałów o konserwacji i modyfikacjach systemów.

Ważne zjawisko opisuje Woodger [14]. Stosując metodę programowania strukturalnego do projektowania i konstrukcji programów, podejmuje się decyzje na wszystkich poziomach abstrakcji. Zwykle niektóre z tych decyzji, nawet na najwyższych poziomach są już opisane w ich końcowej językowej postaci, np. w postaci **if** (jeśli) ... **then** (to) lub **while** (gdy) ... **do** (wykonaj), i pozostają prawie niezmienione w końcowym bezpośrednim tekście programu. W konsekwencji końcowy program składa się z instrukcji ustalonych na wielu poziomach projektowania. Zakładając, że instrukcje pochodzące z różnych poziomów abstracji pisane są w różnych kolorach okazałoby się, że program jest bardzo kolorowy.

Z samej istoty zmian lub poprawek programu wynika, że dotyczą one kilku, a najczęściej tylko jednego poziomu. W „kolorowym" programie tego rodzaju zmiany można zrobić z większym prawdopodobieństwem poprawności niż w programie „czarnym". Możemy wypowiedzieć to także w ten sposób: dobry programista w trakcie modyfikowania programu pamięta stale o różnych poziomach abstrakcji różnych instrukcji. Czytanie cudzych programów jest dlatego tak straszliwie uciążliwe, ponieważ w naszych językach nie można przekazywać poziomów abstrakcji. Współczesne języki są jednopoziomowe. Pewną pomocą może jednakże być mądre użycie deklaracji i wywołań procedur. Świadomie lub podświadomie bardzo wcześnie podejmowano wysiłki, aby chociaż w części uniknąć pomieszania poziomów w dziedzinie diagnostyki błędów. Już w 1960 r. [23] oraz w 1962 r. [24], [25], powstały systemy nieinterpretacyjne, które przekazywały informacje o błędach wykonania na „poziomie języka źródłowego". Jest to proste i dość silne narzędzie, któremu jeszcze dotychczas nie poświęca się dostatecznej uwagi i które nie jest tak powszechnie stosowane, jak na to zasługuje.

Oto inna uwaga historyczna: na początku lat sześćdziesiątych rozgorzał zaciekły spór o zalety parametrów wywoływanych przez wartość w ALGOLu 60. Patrząc z perspektywy czasu, wyglądało to na walkę z duchami. Żadna strona nie zdawała sobie świadomie sprawy z wymagań, takich jak maksimum statycznego wglądu, lokalnej zrozumiałości itp. Zamiast powoływać się na zasady (które wtedy nie były jeszcze wypracowane) typowym argumentem w sporach była umiejętność implementacji tej lub innej „użytecznej metody".

Przez wiele lat trwała żywa działalność w dziedzinie tak zwanych „języków do programowania systemowego" lub „języków zorientowanych maszynowo (wyższego rzędu)", (np. [26], [27]). Czy te języki przyczyniły się do lepszego sposobu projektowania i konstruowania systemów? Niektóre z nich miały interesujące mechanizmy (nowe lub na nowo przypomniane), takie jak oddzielenie dostępu do danych od reprezentacji danych [39], [40], nowe schematy dostępu do danych [41], umożliwienie i zachęcenie do programowania zagnieżdżonego [40], [42], bezpieczne wyskoki na niższy poziom językowy [43] itp. Niewątpliwie działalność ta ożywiła trochę, będącą wówczas w stagnacji dziedzinę projektowania języków ogólnego przeznaczenia. Jednakże idealny język, dostarczający dla każdego wyobrażalnego poziomu abstrakcji właściwych i dobrych operacji, dobrych struktur danych, odpowiednią notację, która pozwoliłaby, w dostatecznym zakresie na komunikację pomiędzy poziomami, która byłaby „kolorowa" w podanym wyżej znaczeniu i która..., dotychczas nie powstał i wszystko wskazuje na to, że nie ujrzy on światła dziennego w najbliższej przyszłości. Opracowanie takiego języka jest zadaniem trudnym, długim, a jednocześnie śmiałym. Możemy nazwać je perspektywicznym planem rozwoju języków, których brak jest przez wiele lat wyraźnie odczuwalny.

Nie jest wcale śmieszne, że producenci komputerów w przeszłości, a nawet obecnie uważają projektowanie i do-

starczanie języków za działalność przeszkadzającą w sprzedaży elektroniki. Sumy pieniędzy przeznaczone przez nich na długoterminowe programy rozwoju technologii produkcji sprzętu w porównaniu z pieniędzmi, jakie przeznaczają na rozwój języków (nie biorąc już pod uwagę planowania perspektywicznego języków) są w dużej dysproporcji z czasem życia odpowiednich produktów. Prawie nikt nie pracuje już na maszynach IBM 704 lub 7090, lecz armie programistów mają wciąż do czynienia z FORTRANem. Trzeba powiedzieć, że lekceważnie języków, które są jedynym środkiem komunikacji pomiędzy człowiekiem i komputerem, które używane są przez setki i tysiące ludzi w rosnącej wciąż liczbie gospodarczo i społecznie ważnych zastosowań jest działaniem nieodpowiedzialnym.
W organizacjach, takich jak duże przedsiębiorstwa lub jednostki administracji państwowej dzięki wieloletniemu doświadczeniu wypracowano schemat działalności, tj. sposób, w jaki zbiera się, przetwarza i rozprowadza do zainteresowanych jednostek odpowiednie informacje. Zwykle istnieje pewna hierarchia jednostek, niektóre odpowiedzialne są za gromadzenie danych, wstępne przetwarzanie i wysyłanie raportów zawierających ostatnio zebrane informacje do określonych innych jednostek.
Istnieją także jednostki, które wyprowadzają na zewnątrz efekty pracy. Jeżeli postawiona wysoko w hierarchii jednostka podejmie decyzję o wykonaniu jakiegoś zadania na podstawie własnych informacji lub danych przesyłanych na żądanie z innych jednostek, to zwykle decyzja ta powoduje podjęcie określonego działania przez inne podległe jednostki. Charakterystyczną cechą takich organizacji jest to, że określone dane „należą" zawsze do organizacji jednostki i operacje na tych danych może wykonać tylko ta określona jednostka (np. lista płac, dane statystyczne). Jest to system „biurowy", w którym każdy urzędnik ma swoje zastrzeżone pole działalności.

Upłynęło prawie ćwierć wieku zanim świadomie zaprojektowano i wprowadzono do dziedziny komputerów sposób uporządkowania danych i procedur, jaki opisano powyżej. Pierwszą pomocną konstrukcją językową było wprowadzenie **OWN** w ALGOLu 60. Znaczny postęp uzyskano przez wprowadzenie pojęcia klasy takiej, jak w SIMULI 67 [31], pojęcia monitora użytego przez Hoare'a [32] i innych. Pojęcia takie mogą też być użyteczne przy szukaniu lepszego, funkcjonalnego uporządkowania w procesie sekwencyjnym (np. [33]) oraz, na koniec, dostarczają nam lepszego zrozumienia co to jest „moduł".

Na prowadzenie tego rodzaju prac w przyszłości wywrze pewien nacisk fizyczne oddzielenie składników elektroniki tworzących jeden system.

Nauczanie programowania systemowego

Przegląd nauczania programowania systemowego należy przeprowadzić w szerszym kontekście nauczania informatyki. Toczą się prawie ideologiczne dyskusje, gdy podejmuje się decyzje, czy prowadzić naukę według zasady „od szczegółów do ogółu", czy też odwrotnie. Autor uważa, że oba skrajne stanowiska stosowane pojedynczo są tak samo złe i bezużyteczne. Wydaje się, że praktycznym jest następujący sposób, wzbogacony dodatkowo doświadczeniami praktycznymi.

Z pedagogicznego punktu widzenia, warunkiem przyjęcia na kurs programowania systemowego powinno być ukończenie, nie związanego być może bezpośrednio, wstępnego kursu, który obejmie np. cały materiał punktu „Wstęp do informatyki", z przytoczonego na wstępie „Programu 68". Wstępny kurs musi być uzupełniony przez intensywne ćwiczenia z programowania z użyciem jednego z lepszych języków wyższego rzędu. Ćwiczenia pomocne są w zdobyciu doświadczenia i rutyny w projektowaniu rozwiązań algorytmicznych dla zadanych problemów. Zwykle wstępny kurs będzie bardzo łagodną formą metody „od ogółu do szczegółów" uzupełnioną tam, gdzie to konieczne przez prezentację typu „od szczegółów do ogółu".
Pierwszy kurs z węższej tematyki programowania systemowego powinien odpowiadać rozwojowi historycznemu, wprowadzać wszystkie podstawowe metody, notacje i pojęcia w sposób „od szczegółów do ogółu". Użycie kilku hipotetycznych, coraz bardziej złożonych maszyn pozwoli słuchaczowi lepiej zrozumieć materiał kursu i umożliwi pracę na ćwiczeniach z przykładami języków maszynowych i assemblerów, stosowanie elementarnych metod programowania do budowy małych translatorów, interpretatorów, funkcji dostępu do danych, podprogramów reagujących w ograniczonym zakresie na przerwania, elementarnych narzędzi służących do synchronizacji itp. Można podkreślić następujące cele takiej metody:
1. Umożliwienie słuchaczowi zapoznania się z typowymi obiektami maszynowymi i operacjami na nich oraz pozwolenie mu na zdobycie pewnej wprawy w tym zakresie. Słuchacz bardzo chętnie podejmie tę „prawdziwą" pracę, mając uprzednio na wstępnym kursie, tylko pośredni kontakt z maszyną poprzez języki wyższego poziomu. Zrozumienie „jak to wszystko działa" jest dla niego przewartościowaniem pojęć.
2. Ukazanie słuchaczowi potrzeby stosowania pojęć wyższego rzędu. Programowanie na poziomie kodu maszyny wiąże się np. z klęską błędów, niezrozumiałością, nieelastycznością kodu maszynowego. Słuchacz powinien doświadczyć negatywnych aspektów ogólności i korzyści płynących z pracy w ramach pewnych dobrze dobranych ograniczeń nawet, gdy może to spowodować chwilowe kłopoty.
3. Przekonanie słuchacza o potrzebie studiowania języków, struktur danych, teorii grafów i innych. Dla większości słuchaczy przekonanie takie jest bezcenne.

Jeżeli mając to przekonanie zaczną zaglądać do książek, np. o językach formalnych, zrozumieją natychmiast „zastosowanie" lub „implementacje" teorii, o których przeczytali. Temat traci swój abstrakcyjny charakter, jest lepiej i szybciej przyswajalny. Jest to główna korzyść z połączenia metody indukcyjnej z dedukcyjną. Z pedagogicznego punktu widzenia, metoda „od ogółu do szczegółów" może zawieść, jeśli słuchacze nie będą znać „szczegółów".
Wszystkie następne kursy programowania systemowego takie, jak konstruowanie kompilatorów, systemy operacyjne, systemy baz danych, konstruowanie systemów mogą w istotny sposób zależeć od wiedzy słuchaczy, wprawy oraz intuicji zdobytych na poprzednim kursie. Na tych kursach nie ma już wówczas potrzeby do przerywania toku wykładu i tłumaczenia „początkowego materiału", Słuchacze są przygotowani do prezentacji materiału z wyższego poziomu metodą „od ogółu do szczegółów" (jeżeli pozwala na to rozwój danej dziedziny) i byłoby grzechem nie wykorzystać tego. Słuchacze mogą ukończyć te kursy posiadając wysoki stopień wiedzy i niezależny sąd.
Nauczanie programowania systemowego jest jedną z działalności, które najbardziej nadają się do przekazania słuchaczowi wyczucia, co to znaczy rozwiązywanie złożonych problemów organizacyjnych, jakie są korzyści, gdy próba zakończyła się korzystnie, a jakie straszne są następstwa porażki.

Nierozwiązane zagadnienia

Chociaż prawie wszystkie zagadnienia, które tu wymieniliśmy były opracowywane w ciągu wielu lat, a w niektórych osiągnięto postęp jesteśmy często daleko od powszechnego wykorzystania ich wyników w przemyśle i w ośrodkach obliczeniowych:

1. Nie istnieje projekt (nie mówiąc już o implementacji) kompilatora kompilatorów, który mając podany na wejściu opis syntaktyczny i sementyczny języka, opis danego systemu operacyjnego oraz ograniczenia stawiane kompilatorowi i jego programom wynikowym wyprodukowałby konkurencyjny, z handlowego punktu widzenia, kompilator. Obecne produkty kompilatorów są często lepsze, jeśli porównujemy niezawodność, lecz mają znacznie gorsze wszystkie pozostałe parametry pracy.

2. Nie prowadzi się ani nie planuje długofalowych prac nad rozwojem języków. Wydaje się, że nikomu na tym nie zależy (przykładem lub bezpośrednią konsekwencją jest to, że nie mamy jasnej koncepcji pośrednich języków wynikowych).

3. Konsekwencję punktu 2 stanowi fakt, że prace w dziedzinie przenoszalności programów gubią się w cząstkowych podjęciach.

4. Zwróćmy uwagę na bardzo ważne, szkodliwe zjawisko w językach programowania. Albo ograniczenia są za bardzo krępujące, albo istnieje potencjalne niebezpieczeństwo klęski lub trzeba zaakceptować znaczne koszty w czasie wykonania.

5. Jak już wspomniano współczesne języki programowania nie posiadają możliwości giętkiego grupowania danych i odpowiadających im operacji, całkowicie bezpiecznych dla każdego przypadku grupowania. Np. większość współczesnych języków nie pozwala na łączenie danych i operacji

w taki sposób, aby dane mogły być przetwarzane tylko za pomocą zadeklarowanego zbioru operacji.

6. Nie mamy żadnych sposobów reakcji na błędy, które z jednej strony byłyby wystarczająco szczegółowe, a z drugiej strony — całkowicie poprawne. Nawet w dziedzinie kompilatorów nie posiadamy automatycznych środków do generowania funkcji reakcji na błędy takich, aby generowane funkcje były chociaż w przybliżeniu równie dobre, jak szczegółowe i indywidualne podprogramy reakcji na błędy pisane ręcznie. Sytuacja jest znacznie gorsza w dziedzinie systemów operacyjnych.

7. Jedną z głównych przesłanek programowania strukturalnego było spostrzeżenie, że o wiele lepiej jest pisać od samego początku programy poprawne. Do sformułowania tej zasady przyczyniły się też obserwacje wyników generatorów automatycznych. Wydaje się, że w literaturze nie ma opisów automatycznego generowania systemów operacyjnych. Czy prace w tym kierunku nie dałyby nowego i niezależnego spojrzenia na problemy konstrukcji systemów operacyjnych?.

Podsumowanie

Pomimo wielu wspaniałych metod i wartościowych osiągnięć indywidualnych, dziedzina programowania systemowego rozwijała się i nadal rozwija się chaotycznie. Jednakże sytuacja się poprawia. Możemy wskazać na pewne oznaki, które uzasadniają nadzieję, że programowanie systemowe stanie się w przyszłości dziedziną nauki.

Podziękowania

Autor zobowiązany jest członkom Grupy Roboczej 2.3. IFIP-u d/s Metodologii Programowania za wiele owocnych dyskusji. Jest także wdzięczny PP J. Burgess i K. Lagally z projektu BSM oraz P C. Corell, H.-G. Hegering i F. Peischl z Leibniz-Rechenentrum z Monachium za krytyczne uwagi o pierwotnej wersji rękopisu.

Tłum. i oprac. **Marek Misala**

REFERENCES

[1] A. C. Shaw, Lecture Notes on a course in systems programming, Technical Report No. 52, Stanford University, December 1966.

[2] Curriculum 68, **Communications of the ACM**, Vol. 11, No. 3, March 1968, 151—197.

[3] J. J. Donovan, **Systems programming**, McGraw-Hill, 1972.

[4] G. Seegmüller, **Einführung in die Systemprogrammierung**, Bibliographisches Institut, Mannheim, 1974 (German).

[5] F. L. Bauer (editor), **Advanced course on software engineering**, Lecture Notes No. 81, Springer, 1973.

[6] Seegmüller, Systems programming as a discipline, Proceedings of the **IFIP Conference on Programming Teaching Techniques**, Zakopane, Poland, September, 1972,North-Holland, 1973.

[7] P. Brinch Hansen, Concurrent programming concepts, **Computing Surveys**, Vol. 5, No. 4, December 1973, 223—245.

[8] G. M. Jenkins, P. V. Youle, **Systems engineering**, Watts, London, 1971.

[9] A. Koestler, **The ghost in the machine**, Hutchinson, London, 1967.

[10] E. W. Dijkstra, The structure of the "THE" — multiprogramming system, **Communications of the ACM**, Vol. 11, No. 5, May 1968, 341—346.

[11] E. W. Dijkstra, Notes on structured programming, **Structured Programming**, Academic Press, 1972, 1—82.

[12] F. W. Zurcher, B. Randell, Interative multi-level modelling — A methodology for computer system design, **Proceedings IFIP Congress 68**, North-Holland, 1968, booklet D, 138—142.

[13] N. Wirth, Program development by stepwise refinement, **Communications of the ACM**, Vol. 14, No. 4, April 1971, 221—227.

[14] M. Woodger, On semantic levels in programming, **Proceedings IFIP Congress 71**, North-Holland, 1971, booklet TA-3, 79—83.

[15] R. A. Snowdon, PEARL: An interactive system for the preparation and validation of structured programs, Technical Report No. 28, University of Newcastleupon-Tyne, Computing Laboratory, November 1971.

[16] N. Wirth, The programming language PASCAL, **Acta Informatica**, Vol. 1, No. 1, 1971, 35—63.

[17] A. N. Habermann, Critical comments on the programming language PASCAL, **Acta Informatica**, Vol. 3, No. 1, 47—57.

[18] N. Wirth, From programming techniques to programming methods, **International Computing Symposium 1973, Davos**, North-Holland, 1974, 47—54.

[19] R. Schild, Interactive structured programming, **International Computing, Symposium 1973, Davos**, North-Holland, 1974, 81—84.

[20] P. A. de Marneffe, D. Ribbens, Holon programming, **International Computing, Symposium 1973, Davos**, North-Holland, 1974, 67—71.

[21] R. Bosch, D. Grune, L. Meertens, ALEPH, a language encouraging hierarchy, **International Computing Symposium 1973, Davos**, North-Holland, 1974, 73—79.

[22] U. Ammann, The method of structured programming applied to the development of a compiler, **International Computing Symposium 1973, Davos**, North-Holland, 1974, 93—99.

[23] J. H. Myers, (private communication), IBM Corporation, Ponghkeepsie, November 1967.

[24] G. Seegmüller, Some remarks on the computer as a source language machine, **Proceedings IFIP Congress 62**, North-Holland, 1962, 524—525.

[25] R. Bayer, D. Gries, M. Paul, H. R. Wiehle, The ALCOR Illinois 7090/94 post mortem dump, **Communications of the ACM**, Vol. 10, 1967, 804—808.

[26] **SIGPLAN-Notices**, Vol. 6, No. 9, October 1971.

[27] **SIGPLAN-Notices**, Vol. 8, No. 9,September 1973.

[28] F. T. Baker, System quality through structured programming, **Proceedings, Fall Joint Computer Conference**, Vol. 41, pt. 1, 1972, 339—343.

[29] J. W. Atwood (editor), Project SUE status report, Technical Report No. 11, University of Toronto, CSRG, April 1972.

[30] G. Goos, J. Jürgens, K. Lagally, The operating system BSM, viewed as a community of parallei processes, Technical Report No. 7208, Rechenzentrum der Technischen Universität München, 1972.

[31] O. J. Dahl, B. Myrhaug, K. Nygaard, **SIMULA 67 common base language**, Norweglan Computing Center, Oslo, 1968.

[32] C. A. R. Hoare, Monitors: An operating system structuring concept, Woorking paper IFIP WG 2,3, Munich, April 1973.

[33] G. Goos, Zum Begriff des Programm-moduls, Interner Bericht Nr. 1/74, Universität Karlsruhe, Institut für Informatik II, 1974, (German).

[34] S. N. Zilles, Procedural encapsulation, a linguistic protection technique, **SIGPLAN — Notices**, Vol. 8, No. 9, September 1973, 143—146.

[35] B. H. Liskov, L. A. Smith, SPIL, a language for construction of reliable system software, **SIGPLAN-Notices**, Vol. 8, No. 9, September 1973, 100—103.

[36] E. W. Dijkstra, The humble programmer, 1972 ACM Turing Award Lecture, **Communications of the ACM**, Vol. 15, No. 10, October 1972, 859—866.

[37] C. A. R. Hoare, An axiomatic basis for computer programming, **Communications of the ACM**, Vol. 12, No. 10, October 1969, 576—581.

[38] C. A. R. Hoare, Proof of a program: FIND, **Communications of the ACM**, Vol. 14, No. 1, January 1971, 39—45.

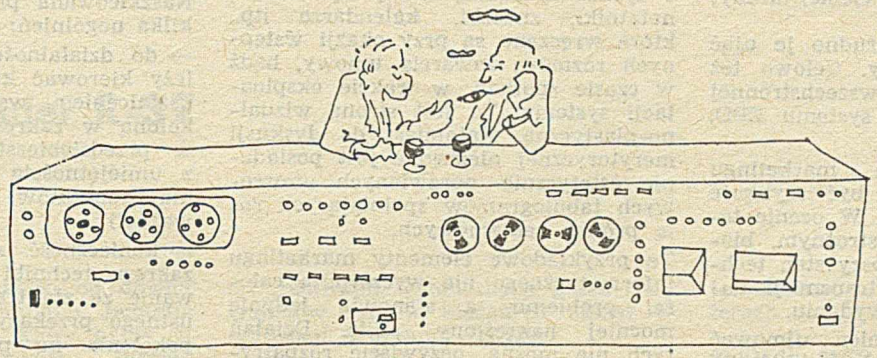
[39] J. D. Ichbiah, J. P. Rissen, J. C. Heliard, The two-level LIS system implementation language, **SIGPLAN-Notices**, Vol. 6, No. 9, September 1973, 79—81.

[40] W. A. Wulf, D. B. Russel, A. N. Habermann, BLISS, a language for systems programming, **Communications of the ACM**, Vol. 14, No. 12, December 1971, 780—790.

[41] L. Mor, J. H. Yott, H. Zima, PROGRESS, a programming language for real-time systems, **SINGPLAN-Notice**, Vol. 7, No. 5, June 1972, 4—24.

[42] B. L. Clark, J. J. Horning, The system language for project SUE, **SIGPLAN-Notices**, Vol. 6, No. 9, October 1971, 79—85.

[43] B. Krieg, H. Scheidig, Language layers portability and program structuring, IFIP/TC-2 Working Conference on Machine Oriented Hugher Level Languages, Trondheim, August 1973.



O marketingu informatycznym słów kilka

Marketing w swoim pierwotnym znaczeniu określał zespół czynników, które w sposób istotny wpływają na realizację sprzedaży i to sprzedaży z zyskiem. To sformułowanie, ulegało ciągłej ewolucji, by nie rzec doskonaleniu — w miarę rozwoju gospodarczego i stopniowego nasycenia rynku masą towarową.

Obecnie marketing koncentruje się na „rynku odbiorcy (konsumenta)", a nie „rynku dostawcy".

Co to oznacza? Przede wszystkim to, iż producent w swoich zamierzeniach wytwórczych powinien sondować opinię przyszłego nabywcy m.in. w zakresie potrzeb i użyteczności danego artykułu (tu ukłon w stronę analizy wartości). Czyli innymi słowy produkować to, czego chce ostateczny nabywca i tym samym dążyć do jego maksymalnego zadowolenia. Trzeba więc mieć wyczucie potrzeb rynku i popytu.

Nie wdając się w całą teorię marketingu, warto podkreślić niektóre jego elementy: funkcjonalność produktu, opakowanie, cena, transport, reklama, dogodne formy sprzedaży. Poza tym, jakość, aspekty psychologii odbiorców, przyzwyczajenia klienta, zespół „public relations".

W informatyce poszczególne elementy tej działalności nie występują w jednakowym natężeniu, jak w dziedzinie sensu stricto handlowej. Nie mniej, wiele z nich jest zbieżnych.

Tu mała dygresja. Informatyka — jako forma przetwarzania i przekazywania informacji za pomocą środków technicznych (głównie komputerów), minęła wprawdzie — w naszych warnkach — okres raczkowania, lecz nie stała się jeszcze powszechna. I właśnie dlatego marketing nie tylko może, ale i powinien znaleźć tutaj szerokie zastosowanie.

W jakich kierunkach winien iść marketing informatyczny?

Analiza tego specyficznego rynku usługowego wskazuje, iż najbardziej palącą sprawą jest efektywność systemów EPD. Przez to należy rozumieć zarówno użyteczność informacji od strony odbiorcy, z całą gamą — niewymiernych — efektów pośrednich, jak i koszty eksploatacji.

Na tym gruncie marketing winien inspirować zapotrzebowanie na „produkty" informatyczne z uwzględnieniem wspomnianych aspektów. Zresztą ta inspiracja winna być zwrotna: ośrodek obliczeniowy (kształtowanie profilu usług) — użytkownik (wyzwalanie potrzeb EPD). Środkami do wywołania tego zapotrzebowania w przedsiębiorstwie są m.in.:

— rozmowy z dyrekcją na temat ogólnych możliwości środowiskowego zastosowania ETO

— robocze spotkania ze średnim dozorem techniczno-ekonomicznym omawiające celowość wprowadzenia zmian (b. ważne, gdyż to oni na codzień mozolą się nad „papierkami"). Ich przecież opinia o EPD i w wyniku tego nowy styl pracy — świadczą o końcowym efekcie użyteczności systemu.

— podkreślanie korzyści wdrożenia analogicznego systemu (podsystemu) w zakładzie o podobnym profilu produkcyjnym

— zapewnianie użytkownika o stałej opiece oraz doradztwie organizacyjnym, prowadzonym przez wysokokwalifikowaną kadrę informatyków i to zarówno w okresie wdrożenia, jak przez cały okres stosowania nowej techniki obliczeniowej. I ten moment jest również bardzo ważny, z uwagi na naturalne bariery psychologiczne ludzi stojących przed każdą nowością — sygnalizowanie przyszłemu użytkownikowi możliwości adaptacji do swojej specyfiki już eksploatowanych gdzie indziej systemów (podsystemów), co znacznie zmniejsza nakłady, nie mówiąc o szybkości wdrożenia.

Wszystko to łatwo napisać, a jak realizować? Pokuśmy się o podanie, na użytek zainteresowanych, kilku przesłanek takiego postępowania:

— postaraj się o nienaganny wygląd

— przed przystąpieniem do rozmów, zapoznaj się dokładnie z organizacją, zakresem działania służb i profilem produkcyjnym przyszłego użytkownika systemu EPD

— nigdy nie przerywaj wywodów swemu rozmówcy

— wykaż szeregowemu pracownikowi, iż nie obcy ci jest tradycyjny system pracy, w ramach tematu przewidzianego do automatyzacji

— w trakcie rozmów, staraj się wyłuskać — przynajmniej jeden, dwa — bezsporne efekty mechanizacji i umiejętnie powracaj do nich (kształtowanie podświadomości). Wkrótce okaże się, iż sam użytkownik pomnoży je wielokrotnie, poprzez wewnętrzne zmiany organizacyjne

— traktuj każdy sygnał (pismo, telefon) użytkownika jako b. ważny. Załatw to szybko i natychmiast daj mu znać. Przez to osiągniesz miano solidnego, rzeczowego i wywołasz wrażenie, że ośrodek obliczeniowy w pierwszym rzędzie chce uwzględniać jego interesy

— mając dobrą „public relations" z użytkownikiem bądź jak najdłużej jego „opiekunem" (przyzwyczajenie i zaufanie klienta), utrzymując z nim ścisłe kontakty

— w sprawach drażliwych zachowaj cierpliwość i pozwól wypowiedzieć się użytkownikowi. Staraj się przyznać mu rację, choćbyś tak bardzo nie był przekonany. Dopiero potem przedstaw uprzejmie i rzeczowo swoje zadanie

— pamiętaj, że w momencie wdrożenia użytkownik daje ośrodkowi obliczeniowemu (poprzez ciebie) kredyt zaufania.

To tylko przykłady i trudno je ująć w jakieś sztywne ramy. Celowo też pomijam obowiązek wszechstronnej znajomości wdrażanego systemu EPD, bo to sprawa ewidentna.

Kolejnym argumentem marketingu informatycznego winna być szybkość przesyłanych informacji. W ocenie tego należy być raczej ostrożnym, biorąc pod uwagę nienajlepszy stan techniczny sprzętu i brak transmisji danych w powszechnym wydaniu.

Jaką jeszcze treść winien ujmować ten specyficzny rodzaj aktywizacji sprzedaży informatycznej? O ile opakowanie nie odgrywa tak istotnej roli jak np. przy sprzedaży artykułów spożywczych, o tyle niezbędna jest stała troska o konserwację (aktualizację) eksploatowanego systemu w wyniku nowych zarządzeń prawno-organizacyjnych. A więc obrona przed naturalnym starzeniem się EPD. Służba markteingowa musi śledzić to zagadnienie, służąc doradztwem zarówno użytkownikowi, jak i projektantom i programistom.

Poza tym, rzeczą ze wszech miar pożądaną jest utrzymywanie ścisłego kontaktu z nowopowstającymi służbami informatycznymi u użytkownika oraz informowanie ich o wszelkich nowościach oprogramowania i sprzętu danego ośrodka obliczeniowego.

Nie można oczywiście pominąć reklamy, jako olbrzymiej siły oddziaływania na odbiorcę ETO. Środkami reklamy winny być zarówno zwięzłe biuletyny, ilustrowane prospekty, wykłady popularyzatorskie, środki masowego przekazu, jak i wszelkiego rodzaju targi komputerowe, zjazdy itp. Ponadto, drobne upominki reklamowe z dyskretną nazwą dawcy np.: teczki, notatniki, znaczki, kalendarze itp., które wręczane są przy okazji wstępnych rozmów, zawarcia umowy, bądź w czasie spotkań w trakcie eksploatacji systemu. To jest strona wizualno-plastyczna. Natomiast do dyskusji merytorycznej niezbędne jest posiadanie estetycznie oprawionych wzorcowych tabulogramów spełniających rolę próbek reklamowych.

Te przykładowe elementy marketingu informatycznego nie wyczerpują całości problemu, a stanowią jedynie mocniej nakreślony szkic. Działań tych nie można oczywiście rozpatrywać w oderwaniu od własnego ośrodka obliczeniowego. Powinny one być sprzężone na całej drodze „od producenta do konsumenta", z rozeznaniem analizy rynku informatycznego dla własnych potrzeb: rozeznanie softwa-re'u, hardware'u w kraju i zagranicą, opinią o sprzęcie, nawiązywaniem kontaktów z innymi ośrodkami obliczeniowymi itp.

We współczesnym marketingu poszczególne sposoby oddziaływania na rynek należy stosować jednocześnie. Problem polega na tym, by dotrzeć do adresata wszelkimi możliwymi sposobami i wpływać nań bezustannie. Dotyczy to również i informatyki.

Spróbujmy hierarchicznie usystematyzować kierunki marketingu informatycznego:

— efektywność systemów EPD
— szybkość dostarczania danych
— konserwacja systemów EPD
— stałe kontakty z użytkownikiem
— cena usług
— aktywizacja sprzedaży usług i reklama
— podtrzymywanie „public relations"

Naszkicowana problematyka, nasuwa kilka uogólnień:

— do działalności marketingowej należy kierować kadrę z wyższym wykształceniem, wszechstronnie przeszkoloną w zakresie EPD, z praktyką w przedsiębiorstwach produkcyjnych, z umiejętnością łatwego nawiązywania kontaktów z użytkownikiem (b. ważne!)

— konieczność stałego doszkalania z zakresu techniki marketingowej. Zerwanie ze złą tradycją chałupnictwa i ustnego przekazywania wiadomości „z pokolenia na pokolenie", na rzecz długofalowego szkolenia kursowego
— wzajemna wymiana doświadczeń tej grupy informatyków — specjalistów (zjazdy, konferencje).

Powyższą wypowiedź oparłem głównie o długoletnie doświadczenia ze współpracy z różnymi przedsiębiorstwami przy rozpowszechnianiu systemów EPD. Bliżej zainteresowanym sprawą marketingu informatycznego podaję następującą lekturę:

— Butler R. S.: Selling, Buying and Shopping Methods, New York 1911
— COINVERSE P. D., Huegy H. W., Mitchell R. V.: Elements of Marketing London 1959
— Tokarska B.: Metody badania i analizy rynków. Warszawa 1964.

Kazimierz Martyniak

Polski komputer R-32

Świat polskiej informatyki od dawna czekał na krajowy komputer Jednolitego Systemu (RIAD), który byłby w stanie optymalnie zaspokoić — wspólnie z komputerami innych krajów socjalistycznych — aktualne i rosnące potrzeby przetwarzania informacji w zakresie planowania i zarządzania gospodarką, automatyzacji prac zawodowych i innych.

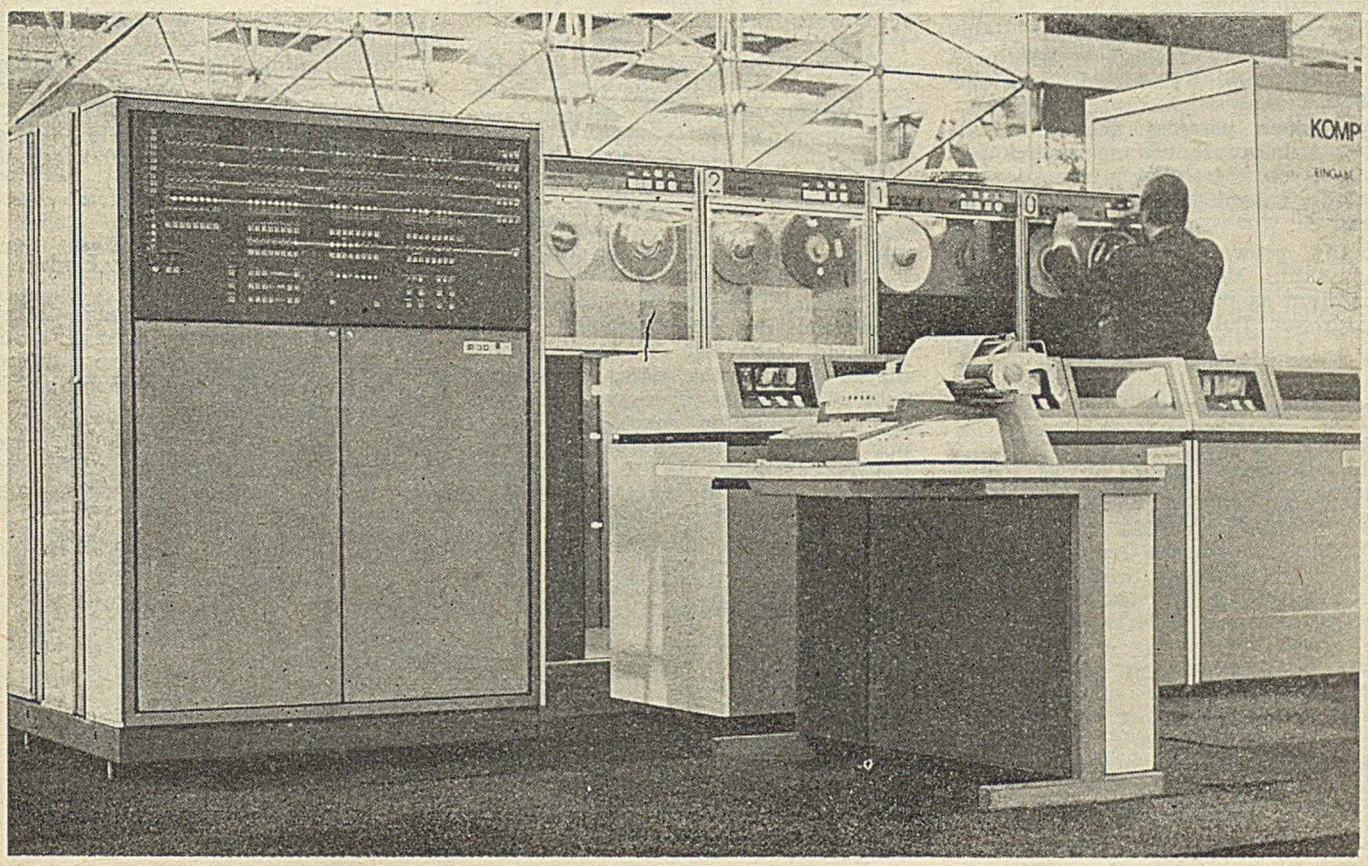
Większość zainteresowanych specjalistów wiedziała o przygotowywanym w MERA-ELWRO komputerze roboczo nazwanym R-30U (lub po prostu R-30) i o technicznej procedurze przyjęcia wyrobu do rodziny RIAD. Dzisiaj możemy już poinformować, że te wieloletnie starania zostały w marcu 1975 r. uwieńczone oficjalnym przyjęciem — polskiego komputera do rodziny RIAD i nadaniu mu szyfru JS-1032 (lub krótko R-32) (po szczegółowym przeanalizowaniu dokumentacji rozwiązań technicznych przez specjalistów krajów biorących udział w realizacji programu RIAD).

Warto przy tym zaznaczyć, że **specjaliści krajów socjalistycznych wysoko ocenili rozwiązania logiczne, konstrukcyjne i technologiczne polskiego komputera R-32** oraz jego szczególnie korzystny — cytujemy — „stosunek wydajności przetwarzania do kosztu produkcji".

Maszyna R-32 należy (dzięki swoim parametrom) do średniej klasy współczesnych komputerów. Rozwiązania techniczne tej maszyny pozwalają uzyskać ponad 200 000 tzw. średnich operacji na sekundę lub około 500 000 dodawań na sekundę. Natomiast wielkość sprzętu procesora i kanałów nie przekracza 30 pakietów o wymiarach 29,5 × 15 cm. Czyli tyle, ile przeważnie spotyka się w tzw. minikomputerach lub w bardzo małych komputerach.

Zakłady MERA-ELWRO wyprodukują w roku bieżącym kilkanaście systemów komputerowych R-32. W skład każdego systemu wejdzie: do 6 jednostek pamięci dyskowej JS-5052 (bułgarskie), do 6 jednostek pamięci taśmowej JS-5019 (polskie), jedna lub dwie drukarki wierszowe JS-7033 (polskie) oraz wiele innych urządzeń wejścia-wyjścia. Wszystkie systemy roku 1975 będą od razu wyposażone w system operacyjny DOS/JS oraz w odpowiednie oprogramowanie użytkowe. Dalsze systemy software'owe są w przygotowywaniu i będą dostarczane użytkownikom sukcesywnie.

Thanasis Kamburelis
Andrzej Zasada
OBR MERA ELWRO



Seminarium Naukowe: INFORMATYKA W DYDAKTYCE I BADANIACH

W pierwszej połowie lutego br. Politechnika Wrocławska i TNOiK — Oddział Wrocław zorganizowały w Cieplicach Śląskich seminarium z dyskusjami okrągłego stołu na temat informatyki w dydaktyce i badaniach, ze szczególnym uwzględnieniem systemów wielodostępnych.

Uczestnikami seminarium byli zagraniczni naukowcy i specjaliści w zakresie informatyki: A. Gilmour, R. L. Graves z Chicago, J. Scala Estalela z Madrytu, J. Puzman z Pragi, R. Pořižek z Bratysławy, F. Stuchlik z Magdeburga, T. Szantovanyi i J. Talloczy z Budapesztu, jak również krajowi naukowcy i specjaliści: prof. dr hab. K. Fiałkowski, dyr. mgr inż. J. Kramarczuk, prof. Z. Pawlak, prof. W. Turski, doc. dr inż. St. Waligórski z Warszawy oraz doc. dr inż. M. Bazewicz, prof. dr hab. inż. Z. Bubnicki, dyr. mgr inż. E. Bilski, doc. dr inż. W. Grudzewski, dyr. mgr inż. J. Muszyński z Wrocławia.

Tematem dyskusji seminarium były:
— zagadnienia związane z powszechnym kształceniem specjalistów w zakresie podstaw i zastosowań informatyki
— zagadnienia kształcenia specjalistów informatyków
— wymiana doświadczeń w zakresie metod i technik kształcenia specjalistów w zastosowaniu informatyki w organizacji i zarządzaniu, naukach ścisłych, społeczno-ekonomicznych oraz efektywność metod
— stan i kierunki rozwojowe środków technicznych (sprzęt, oprogramowanie) służących dydaktyce i prowadzeniu badań w systemach wielodostępnych
— przyszłościowe rozwiązania w dziedzinie informatyki z punktu widzenia potrzeb współpracy dialogowej w procesach kształcenia i prowadzenia badań.

Wymieniona powyżej tematyka dyskusji była prezentowana pod kątem światowych trendów rozwojowych wielodostępnych systemów informatycznych.

Obrady prowadzone były w języku angielskim. Materiały z dyskusji seminarium są publikowane w zeszytach naukowych Politechniki Wrocławskiej.

(MB)

Szewc bez butów ...

ZOINTE — placówki utworzone w celu ułatwienia pracy głównym producentom rozstrzygającym o sukcesach ZETO — projektantom i programistom — co tu dużo mówić — nie spełniają swego zadania. To znaczy — nie informują. Swego czasu informatycy w pewnym ZETO biedzili się strasznie nad rozwiązaniem problemu, który rozwiązano już dawno (oprogramowanie firmowe ICL) w innym. Autor osobiście dostarczył dokumentację, bo szczęściem o tym wiedział.

Dzieje się tak na skutek sprzężonych ze sobą (zwrotnie) przyczyn natury zewnętrznej i wewnętrznej. Uzasadnienie pewnej inercji ZOINTE wyglądałoby tak: ponieważ ich motywacje są słabe — możliwości też skromne w stosunku do innych działów pracy ZETO, ponieważ skromne możliwości — skromne wymagania odbiorców, ponieważ skromne wymagania odbiorców, słabe motywacje dla ZOINTE; koło się zamyka.

Przeprowadzona w marcu pod egidą Działowego Ośrodka Informacji OBRI narada kierowników ZOINTE-ZETO unaoczniła smutny fakt, że w drugie dziesięciolecie działalności wielu ZETO, ZOINTE wkraczają bez zautomatyzowanego systemu gromadzenia i wyszukiwania informacji naukowo-technicznej. Szewc bez butów chodzi... Tak właśnie wygląda rzeczywistość. Ale przysłowie powstało ongiś gdy szewc był tak biedny, że robiąc buty nie mógł sobie pozwolić na własne. Tymczasem chyba posiadaczy komputerów stać na luksus nowoczesnego środka do stymulacji własnej działalności.

Nie, nie na pokaz. Oczywiście. Po prostu dla własnych potrzeb. Ewentualny argument, że byłoby to strzelane z armaty do wróbla (małe zbiory informacji — duży koszt eksploatacji systemu) wydaje się do odparcia — zbiory są takie, jakie można tworzyć metodą chałupniczą. Nie znaczy to, że są wystarczająco duże. Mogą, a nawet powinny być większe. Jednakże ich powiększanie jest warunkowane przez możliwości.

W którą stronę powiększać zbiory informacji w ZETO nie czuję się kompetentny orzec. Może nie tylko biblioteki książek, czasopism, druków, może i biblioteki programów, może co innego?

Dość, że sprawa automatyzacji winna być szybko i dobrze załatwiona. Wspomniana konferencja w głównej mierze temu problemowi została poświęcona.

Jak jednak wynikło w dyskusji nie zapowiada się, aby ZOINTE mogły wcześniej korzystać z automatyzacji niż za 6 lat. To za późno. Zwłaszcza, że DOI OBRI zamierza nie mieszkając przystąpić do szkolenia pracowników ZOINTE w sposobach korzystania z systemu MARK (firmy ICL) przewidzianego do katalogowania i gromadzenia informacji w sieci ZETO. Na marginesie: oponenci podnosili (i słusznie), że za lat 6 system ten może być już przestarzały, a więc szkolenie już teraz byłoby trudem zbytecznym.

Przysłuchując się głosom podniesionym na konferencji, nabrałem podejrzeń, że władze ZETO usługowy charakter ZOINTE pojmują zbyt dosłownie i zbyt jednostronnie. To znaczy ZOINTE świadczy dla ZETO; ZETO nie musi dla ZOINTE. Nie muszę chyba podkreślać (gdyby istotnie tak było), że traktowanie swojej placówki, jako zewnętrznej w stosunku do całego organizmu ZETO jest samo w sobie niebezpieczne. Myślę tylko, że na naradzie poznałem sporo osób, które mając możliwość korzystania ze sprawniejszych metod pracy, przybliżyłyby działalność ZOINTE do modelu założonego u podstaw powołania tych placówek w strukturze ZETO. (K. B.)

Oferta dla Czechosłowacji

Przewiduje się, iż w latach 1976—80 gospodarka czechosłowacka będzie musiała uzyskać 750 komputerów. Realizacja tak wielkiego zamierzenia wymaga nie tylko olbrzymich nakładów finansowych, ale również jeszcze ściślejszej współpracy międzynarodowej, zwłaszcza z krajami socjalistycznymi.

Wychodząc naprzeciw tym planom Przedsiębiorstwo Handlu Zagranicznego MERA-METRONEX otworzyło 10 bm. w Pradze wystawę polskiego przemysłu komputerowego. Prezentowane urządzenia w dziedzinie elektronicznej techniki obliczeniowej są nie tylko wizytówką możliwości najnowocześniejszej polskiej techniki, lecz zarazem konkretną ofertą handlową.

Czechosłowacja jest dziś, obok Związku Radzieckiego i NRD, jednym z głównych odbiorców naszych urządzeń komputerowych. W okresie ostatnich 2 lat eksport MERY-METRONEXU do CSRS wzrósł 3,5-krotnie — pracuje tu już m. in. około 60 komputerów serii ODRA 1300. (kr)

Klinika komputerów w Łodzi

MERASERW-3 autoryzowana stacja konserwacji i napraw maszyn systemu MERA, a także aparatów pomiarowych stosowanych w przemyśle lekkim powstała w Łodzi.

Na wyposażenie nowej placówki przeznaczono 5 mln zł. Jeszcze w tym roku w Łodzi będzie pracowało ponad 200 minikomputerów MERA. (el)

Zastosowanie metod matematycznych i informatyki w geologii krajów RWPG

Działająca od 1970 r. a powołana po raz pierwszy w 1966 r., stała grupa robocza RWPG informatyków specjalizujących się w zastosowaniach metod matematycznych i ETO w działalności geologicznej — ustaliła plan pracy na lata 1976—80. Nastąpiło to podczas spotkania grupy w lutym br. w NRD — połączonego z seminarium na temat „Zastosowania metod matematycznych w badaniach geologicznych" zorganizowanego przez Uniwersytet im. E. M. Arndta z Greiszraldu. Podczas spotkania grupy oceniono i podsumowano stan realizacji wspólnie przez kraje członkowskie RWPG opracowywanego w latach 1971—75 problemu „Zastosowania metod matematycznych i ETO w dziedzinie geologii w krajach członkowskich RWPG".

Realizacja tego problemu miała na celu:

— przyspieszenie procesu wdrażania metod matematycznych i ETO do prac poszukiwawczych w geologii krajów RWPG

— zmniejszenie wydatków związanych z opracowaniem nowych algorytmów i programów niezbędnych w procesie rozwiązywania zadań geologicznych.

Cele te, w latach 1971—75 realizowano m. innymi poprzez:

— wymianę informacji dotyczących zastosowań metod matematycznych i ETO w geologii

— organizowanie krajowych bibliotek algorytmów i programów rozwiązywania zadań geologicznych

— wymianę programów pomiędzy zainteresowanymi krajami lub też poprzez udzielanie pomocy przez kraje bardziej zaawansowane w rozwiązywaniu złożonych problemów geologicznych — na prośbę krajów o mniejszym doświadczeniu

— wydawanie informatora o algorytmach i programach (3 edycje w ciągu 5 lat)

— organizowanie seminariów dla specjalistów, podczas których najnowsze rezultaty prac referowali zarówno uczeni z krajów RWPG, jak też i z innych krajów.

Współpraca krajów członkowskich koncentrowała się głównie na rozwiązywaniu zagadnień dotyczących:

— zastosowań metod matematycznych w toku poszukiwań złóż surowców

— zastosowań metod numerycznych w procesie dokumentowania złóż i ich oceny ekonomicznej

— budowy zautomatyzowanych systemów przetwarzania danych geologicznych.

Rezultaty prac i wyniki praktycznych zastosowań w procesie rozpoznawania różnych surowców i kopalin były referowane podczas sympozjów w Ałma-Ata (ZSRR — 1971 r.), Przybrani (CSRS — 1973), Sellin (NRD — 1975 rok). Oceniając obecny stan zastosowań metod matematycznych i ETO w działalności geologicznej należy odnotować wyraźny postęp będący wynikiem zmiany w ocenie możliwości efektywnego wykorzystania tych metod i ETO w geologii. Obecnie można już stwierdzić że metody matematyczne i ETO znalazły zastosowanie praktycznie na wszystkich szczeblach działania służb geologicznych krajów RWPG. Zastosowania tych metod i ETO traktuje się w planach na lata 1976—80, jako konkretną rezerwę możliwości produkcyjnych geologii. Istotnym jest fakt podwojenia liczby komputerów zainstalowanych w organizacjach geologicznych krajów RWPG w stosunku do 1970 r., przy czym obecnie często już można spotkać komputery w geologicznych przedsiębiorstwach produkcyjnych. Podobnie jak i w innych dziedzinach działalności gospodarczej, nie udało się dotychczas ujednolicić parku komputerowego służb geologicznych krajów RWPG. Stan taki obniża efektywność współpracy krajów członkowskich, jednakże pojawienie się na rynku komputerów JS RIAD stworzyło służbom geologicznym krajów RWPG szansę korzystania z jednolitego sprzętu informatycznego, co zapewne nastąpi po 1980 r. Jednocześnie jako godny odnotowania fakt uznać należy uzyskany w ostatnich 2 latach postęp w budowie systemów informatycznych dla różnych dziedzin geologii.

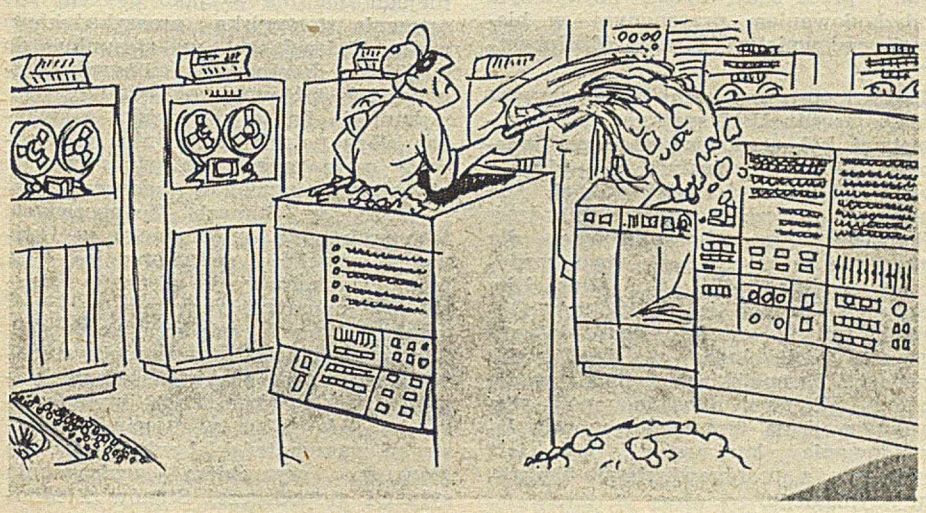
Dążenie do jednolitości sprzętu i postęp w budowie systemów informatycznych — stanowią bazę, na której opracowano plan współpracy służb geologicznych w dziedzinie informatyki, w latach 1976—80.

Problem do wspólnej realizacji w latach 1976—80 określono jako „Opracowanie zautomatyzowanych systemów przetwarzania informacji geologicznych". W ramach problemu wydzielono 2 podproblemy, a to: „Opracowanie zautomatyzowanych systemów obliczania zasobów kopalin użytecznych" i „Opracowanie różnorodnych metod matematycznych rozwiązywania zadań geologicznych w toku prac poszukiwawczych".

Obydwa podproblemy rozwiązywane będą na bazie JS RIAD, a krajami wiodącymi wyznaczono WRL i ZSRR. Wspólnie również zrealizowany zostanie kurs mający na celu podwyższenie kwalifikacji informatyków specjalistów z zakresu optymalizacji ocen zasobów złóż kopalin użytecznych. Istnieją pewne możliwości zorganizowania centrum koordynacyjnego dla realizacji problemu „Opracowanie zautomatyzowanych systemów przetwarzania informacji geologicznych". Grupa robocza zaleciła zorganizowanie krajowych fondów algorytmów i programów zorientowanych na maszyny JS RIAD, a dotyczących geologii i ekonomiki surowców mineralnych.

Zaaprobowano na lata 1976—80, a więc do czasu wyposażenia krajowych służb geologicznych w komputery JS RIAD, dokonanie reorientacji w zakresie języków programowania z powszechnie stosowanego w geologii ALGOLu na FORTRAN IV, COBOL, PL-1 (PLAN), ASSEMBLER, co ma stanowić przygotowanie do stosunkowo łatwego przejścia na JS RIAD.

Udział delegacji polskiej w opracowaniu tego konstruktywnego planu współpracy był znaczny, a propozycje dotyczące form współpracy i kształcenia kadr stały się podstawą szczegółowego programu prac w ramach wymienionych problemów. Warto odnotować fakt przystąpienia do realizacji programu współpracy przez pozaeuropejskie kraje RWPG tj. Kubę i Mongolię, które stawiają pierwsze kroki na niwie komputeryzacji działalności geologicznej i wprowadzania do geologii metod matematycznych.



Maciej K. Radecki

IBM oskarżony o naruszenie amerykańskich ustaw antytrustowych

Pod koniec lutego br. rozpocząć się ma w Waszyngtonie największy proces w historii prawodawstwa Stanów Zjednoczonych przeciwko firmie oskarżonej o naruszenie przepisów antytrustowych.

Departament Sprawiedliwości USA, po prawie sześcioletnim gromadzeniu materiałów, rozpocznie trudny pojedynek z International Business Machines Corporation — gigantem, który obecnie kontroluje 80% amerykańskiego i 60% światowego rynku środków informatyki. Zdaniem wielu obserwatorów, celem Departamentu Sprawiedliwości jest doprowadzenie do rozbicia korporacji na co najmniej dwa odrębne, konkurencyjne koncerny.

Nie będzie to pierwszy proces przeciwko IBM wytoczony firmie na podstawie amerykańskiego prawa antytrustowego. Koncern ten jest, zwłaszcza w ostatnich latach, najczęściej oskarżaną o próby monopolizacji rynku i tłumienie konkurencji firmą w Stanach Zjednoczonych. Wynika to oczywiście ze specyficznego, kapitalistycznego rynku sprzętu informatycznego, na którym doszło do największej (poza wydobyciem i sprzedażą diamentów) koncentracji produkcji, sprzedaży i usług w rękach jednej kompanii. Historia „kontaktów" IBM-u z ustawami antytrustowymi liczy już ponad 60 lat, gdyż to w 1913 roku, założyciel i pierwszy prezes firmy — Thomas J. Watson (senior) — wtedy jeszcze współpracownik National Cash Register — został wraz z 28 swoimi współpracownikami uznany winnym „eliminacji zasady wolnego handlu i stworzenia faktycznego monopolu" przez ławę przysięgłych w Cincinnati.

W okresie międzywojennym oraz w pierwszych latach po Drugiej Wojnie Światowej miało miejsce kilka podobnych procesów, wytoczonych IBM-owi przez jego słabszych konkurentów, jednakże w każdym przypadku stroną wygrywająca był IBM, dysponujący, poza oczywiście rosnącą coraz bardziej potęgą ekonomiczną, dużym sztabem doskonałych prawników, specjalizujących się w zagadnieniach amerykańskiego prawa antytrustowego.

O roli sztabu prawniczego wewnątrz korporacji świadczyć może fakt, iż wieloletnim jego dyrektorem był Nicholas de Katzenbach, były prokurator generalny Stanów Zjednoczonych, który obecnie mianowany został wiceprezesem koncernu.

Szczególnie intensywnie pracuje wydział prawa antytrustowego IBM od roku 1969, gdyż w ciągu ostatnich pieciu lat rozpoczęto i prowadzono kilka poważnych dochodzeń sądowych, zainicjowanych tak przez krajowych konkurentów giganta, jak i przez organa administracji USA i innych krajów. Spośród tych dochodzeń należy przede wszystkim wymienić wspomniane już postępowanie dochodzeniowe wszczęte w styczniu 1969 roku przez amerykański Departament Sprawiedliwości. Departament oskarża IBM o naruszenie postanowień paragrafu 2, tzw. „Sherman Antitrust Act", o świadome dążenie do zmonopolizowania handlu sprzętem elektronicznej techniki obliczeniowej na wewnętrznym rynku USA oraz o wykorzystywanie sieci swych filii zagranicznych w celu omijania przepisów podatkowych.

W kwietniu 1969 roku, XEROX Corporation wniosła do sądu sprawę przeciwko IBM, oskarżając ten koncern o naruszenie swoich praw patentowych i nieprawne przejęcie niektórych poufnych informacji umożliwiających skonstruowanie kopiarki IBM Copier I. Dodać należy, iż we wrześniu 1973 roku XEROX wniósł dodatkowe oskarżenie dotyczące odmiany tego wyrobu — IBM Copier II.

W styczniu 1972 roku rozpoczęło się przed okręgowym sądem w Tulsa dochodzenie z powództwa firmy TELEX Corporation, która oskarżyła IBM o uprawianie praktyk monopolistycznych oraz żądała wyrównania strat poniesionych w związku z tym, ocenianych na 1 084 mln dolarów. Proces w tej sprawie zakończył się we wrześniu 1973 roku przyznaniem firmie TELEX odszkodowania w wysokości 259,5 mln dolarów, które nigdy nie zostały przez IBM zapłacone. W lipcu 1972 roku zakończone zostało postępowanie w sprawie skargi firmy *software'*owej GREYHOUND COMPUTER Corporation o stosowanie przez IBM monopolistycznych cen usług. Wyrok odrzucił wszelkie oskarżenia, sąd bowiem nie dopatrzył się w postępowaniu IBM naruszania obowiązujących zasad.

W listopadzie i grudniu 1973 roku wpłynęło do sądów amerykańskich sześć odrębnych skarg, w związku z domniemanym naruszeniem przez IBM ustawy Shermana. Firmy: CALIFORNIA COMPUTER PRODUCTS INC, TRANSAMERICA COMPUTER CO., MARSHALL INDUSTRIES, HUDSON GENERAL CORP., CENTURY DATA SYSTEMS oraz MEMOREX żądają od IBM wypłacenia im odszkodowania o łącznej wartości 3,8 miliarda dolarów — z tego na kompanię MEMOREX przypada 3 150 mln dolarów — co w przypadku zatwierdzenia tej sumy przez sądy, byłoby najwyższym odszkodowaniem przyznanym w historii sądownictwa Stanów Zjednoczonych.

Także i poza granicami USA poczynania koncernu IBM spotkały się z protestami kół gospodarczych i politycznych, które w ubiegłym roku doprowadziły w dwóch wypadkach do wszczęcia szczegółowych dochodzeń. Najpoważniejsze konsekwencje dla koncernu może mieć śledztwo rozpoczęte w lipcu 1974 roku Brukselską Komisję Europejskiej Wspólnoty Gospodarczej w sprawie zbadania zagadnień nielegalnego podziału rynku zachodnioeuropejskiego i nadużywanie przez IBM swej quasi-monopolistycznej pozycji na tym rynku w celu likwidowania konkurencji, co jest sprzeczne z postanowieniami Traktatu Rzymskiego.

Ponadto rząd federalny Kanady powołał specjalną komisję, która zajmie się zagadnieniami nadmiernego ingerowania kompanii macierzystej w sprawy kontaktów handlowych kanadyjskiej filii IBM. Bezpośrednim powodem tak ostrego wystąpienia władz kanadyjskich było anulowanie przez amerykańską centralę IBM kontraktu na dostawę sprzętu informatycznego na Kubę, zawartego przez filię kanadyjską.

Wracając do mającego się rozpocząć procesu Departamentu Sprawiedliwości przeciw IBM, chciałbym zwrócić uwagę na dwa wydarzenia, które wywołały liczne komentarze na temat ewentualnego wyniku procesu.

Pierwszym z nich było mianowanie 14 stycznia 1975 roku Edwarda Levi, rektora uniwersytetu w Chicago, specjalizującego się w dziedzinie prawa antytrustowego, na stanowisko Ministra Sprawiedliwości Stanów Zjednoczonych. Pośrednim rezultatem tej nominacji może być etatowe wzmocnienie wydziału antytrustowego, który jak dotąd, nie był zdolny do rozpatrywania wszystkich przypadków naruszania obowiązujących przepisów. O skali trudności w pracy wydziału świadczyć może fakt, iż sama tylko IBM przez trzy lata od rozpoczęcia postępowania antytrustowego dostarczyła do Departamentu Sprawiedliwości 87 ton (sic!) różnych dokumentów. Stosunkowo mała obsada osobowa wydziału antytrustowego jest jedną z najpoważniejszych przyczyn ograniczonego zastosowania w praktyce ustaw zwalczających monopolizację gospodarki.

Drugim wydarzeniem, tym razem jednoznacznie świadczyć mającym na korzyść oskarżonej kompanii, był wyrok sądu apelacyjnego w Denver, do którego odwołała się IBM. Sąd ten 24 stycznia 1975 roku anulował wspomniany już wyrok w sprawie TELEX versus IBM, potwierdzając jednocześnie zasadność odszkodowania na rzecz IBM za bezprawne przejęcie niektórych tajemnic kompanii z Armork w wysokości 17,5 mln dolarów.

Podczas gdy nominacja nowego Ministra Sprawiedliwości oceniana jest niejednoznacznie — jako element zaostrzenia stanowiska amerykańskiego rządu federalnego w stosunku do wielkich kompanii monopolistycznych, lub też wprost odwrotnie, jako wysunięcie na to stanowisko człowieka, który potrafi skutecznie „dostosować" ostre przepisy antytrustowe do aktualnych wymogów depresji gospodarczej w Stanach Zjednoczonych, to wyrok sądu apelacyjnego w Denver powszechnie uznawany jest za przejaw zahamowania ofensywy przeciwko monopolistycznym poczynaniom niektórych wielkich korporacji. Jak sądzę, ostatnie wydarzenia stanowią część programu umocnienia pozycji wielkich koncernów przemysłowych USA, które w ujęciu administracji waszyngtońskiej winny stać się głównym czynnikiem uzdrowienia sytuacji gospodarczej Stanów Zjedno-

czonych. Program ten zmierza do wzrostu konkurencyjności firm amerykańskich na rynku światowym oraz ochrony przed zakusami obcych kapitałów (szczególnie pochodzących z naftowych krajów Bliskiego Wschodu), których właściciele, zachęceni ogólną i poważną obniżką notowań akcji na giełdach amerykańskich, dążą do przejmowania kontroli finansowej nad niektórymi sektorami gospodarki USA. Temu celowi służyć ma projekt ścisłego ograniczenia udziału obcych kapitałów w aktywach finansowych firm USA oraz projekt ogólnej obniżki podatków nakładanych na korporacje.

W tym kontekście, sytuacja najbardziej kiedyś dochodowej wielkiej firmy amerykańskiej wydaje się nie być tak zła, jak wydawała się jeszcze w połowie ubiegłego roku, gdy ograniczenie zapotrzebowania na światowym rynku sprzętu informatycznego, połączone ze wspomnianymi poważnymi kłopotami prawnymi, spowodowało najbardziej gwałtowny spadek notowań akcji IBM na światowych rynkach finansowych.

Jerzy Miłek

Kollokwium: szkolenie informatyków w NRD

Sekcja „Ekonomika przedsiębiorstw socjalistycznych" Uniwersytetu Rostock (NRD) zorganizowała w dniu 18.9.74 r. przy współpracy z Radą Naukową d/s Nauk Ekonomicznych przy Ministerstwie Szkół Wyższych i Badań Naukowych NRD Konferencję naukową na temat problemów szkolenia informatycznego na kursie podstawowym nauk ekonomicznych. W czasie konferencji gospodarze przedstawili swoje doświadczenia uzyskane w prowadzonym po raz pierwszy szkoleniu informatycznym studentów wg nowego jednolitego planu studiów ekonomicznych. Głównymi akcentami konferencji były następujące zagadnienia:

— wpływ nauczania informatyki na kształtowanie się osobowości socjalistycznej

— wpływy nauczania informatyki na umocnienie motywacji zawodowej oraz problemy metodyki nauczania.

Ponadto omówiono problemy koordynacji treściowej z korespondującymi kompleksami nauczania, jak również sposoby niezbędnego pogłębiania w fazie specjalizacji kierunkowej wiadomości uzyskanych przez studentów w czasie kursu podstawowego. Dyskusja, w której wzięli udział liczni nauczyciele akademiccy z różnych kierunków studiów doprowadziła do następujących stwierdzeń:

● Nowy plan studiów ekonomicznych z punktu widzenia nauczania informatyki spełnił całkowicie oczekiwanie.

● W oparciu o ten program konieczne jest dalsze rozwinięcie metodyki nauczania oraz silniejsze uwzględnienie działania czynników oddziałujących na kształtowanie świadomości socjalistycznej. Szkolenie w dziedzinie informatyki jest priorytetową częścią szkolenia naukowców ekonomistów.

● Niezbędne jest rozszerzenie problematyki zastosowań informatyki w fazie specjalizacji kierunkowej, co jednak wiąże się z szeregiem trudności obiektywnych i subiektywnych. Trudności te muszą być w najbliższym czasie przeanalizowane i przezwyciężone wspólnie z kompetentnymi specjalistami z tej dziedziny.

Drugi punkt konferencji dotyczył problemu wykorzystania doświadczeń radzieckich w dziedzinie szkolenia informatycznego ekonomistów. Podkreślono szeroki zakres korzystania zarówno z literatury, jak i bezpośrednich możliwości wymiany doświadczeń przez pracowników naukowych i studentów z NRD w czasie staży w Instytucie Budowy Okrętów w Leningradzie i innych radzieckich placówkach naukowych. (WK)

Wg Rechentechnik/Daten Verarbeitung 2/1975

Węgierski model mózgu

Naukowcy instytutu badawczego w Budapeszcie opracowali prototyp automatu komórkowego — maszyny matematycznej, która w najbliższej przyszłości dokona prawdziwej rewolucji w dziedzinie komputerów. Automat jest matematycznym modelem ludzkiego mózgu i zdaniem naukowców będzie w stanie sam tworzyć nowe pojęcia. W swojej pracy węgierscy naukowcy oparli się na pomyśle nieżyjącego już, wybitnego matematyka amerykańskiego węgierskiego pochodzenia J. von Neumanna.

Automat komórkowy działa, podobnie jak ludzkie komórki nerwowe, na zasadzie wzajemnych połączeń i oddziaływujących na siebie miniaturowych elementów czyli neuronów.

Dobiegają już końca prace nad projektami programowania i jednostek obwodu zapisującego automatu. Obecny prototyp zostanie zminiaturyzowany przed skierowaniem do masowej produkcji. Maszyna ta, która ma zawierać 200 tysięcy komórek, będzie się charakteryzowała nieograniczoną ilością połączeń, samoczynną reprodukcją i wielką szybkością. (kr)

Komputer feruje wyrok śmierci

Kiedy umrę? na to pytanie dręczące człowieka od zarania dziejów odpowiada w Kanadzie komputer. Trzeba mu tylko podać szczegółowe informacje, dotyczące m. in. sposobu odżywiania się, picia alkoholu i palenia, przyczyny zgonu przodków, sposobu prowadzenia samochodu, szybkości reakcji itd. „Po przetrawieniu" tych informacji komputer mówi jaka jest długość jego życia z punktu widzenia medycyny oraz ostrzega, jaka choroba i kiedy mu zagraża. Opierając się na danych statystycznych, dotyczących przeciętnej długości życia i przyczyn zgonów w kraju, komputer określa przypuszczalną datę śmierci człowieka, w razie gdyby nie zmienił trybu życia.

Powierzenie komputerowi tej problematyki jest częścią składową kanadyjskiego programu rządowego ochrony zdrowia, który ma na celu przedłużenie życia ludności. Z usług komputera skorzystało już 10 tys. osób. Powstaje tylko pytanie: ilu z nich świadomość rychłego zgonu skróciła życie? (kr)

Wiadomości różne

● Obserwuje się na świecie **dążenie do zwiększania pojemności jednostek dyskowych.** Jednym z ostatnich osiągnięć jest MARK XI — produkcji Memorex Corp. o pojemności 200 milionów bajtów, kompatybilny z modelami MEMOREX 3675 i IBM 3330-11.

Z powodu wymaganych wysokich tolerancji przy zapisie o podwójnej gęstości, powierzchnia robocza dysków jest oparta o specjalnie utwardzany nośnik materiału magnetycznego. Pakiet składa się z 12 dysków, w tym 19 płaszczyzn roboczych, jedna dla obsługi serwomechanizmów przesuwu głowic i po jednej dla ochrony pakietu od góry i dołu. (lj)

● „Sea Saint", **pierwszy z 15 szwedzkich supertankowców wyposażono w rozwinięty system komputerowy** do sterowania statkiem oraz rozładunkiem i załadunkiem ropy. System — wartości ponad 1 milion dolarów — obejmuje między innymi 2 norweskie maszyny cyfrowe Kongsberg SM-4, każda z pamięcią 64 K słów. (lj)

● według oświadczenia Departamentu Handlu USA amerykańscy producenci obwodów scalonych osiągnęli dwuletnią przewagę w zakresie postępu technicznego nad zagranicznymi konkurentami i mają znakomitą pozycję do dalszego zwiększenia tej przewagi.

Zostało to osiągnięte, zdaniem Departamentu, wskutek zmniejszenia w innych krajach funduszy na prace badawczo-rozwojowe oraz przesadną ostrożność firm zagranicznych. (lj)

●TAKACHIHO-BURROUGHS, firma komputerowa, która powstała z połączenia TAKACHIHO KOHEKI (Japonia) z BURROUGHS (USA) rozpocznie sprzedaż nowego systemu EPD w mniej więcej tym samym czasie, gdy IBM udostępni dane swojego nowego systemu FS (Future System), zapowiadanego jako **reprezentanta nowej technologii maszyn cyfrowych.** TAKACHIHOc-BURROUGHS wprowadził takie opracowania, jak VS (Virtual System) i MP (Multiple Processor). (lj)

Decyzja o rozpoczęciu prac nad nowym systemem konkurencyjnym w stosunku do IBM'owskiego FS będzie miała z pewnością wpływ na przemysł maszyn cyfrowych zarówno w Japonii, jak i na świecie.

● DIGITAL EQUIPMENT Corp. rozpoczął produkcję **nowych wersji systemu DEC-10 o nazwie DEC system 1080 i 1090,** które przy zbliżonej cenie mają mieć możliwości dwukrotnie większe od istniejących DEC-10. Leży to w profilu zainteresowań firmy dużymi sieciami z podziałem czasu, zdalnym przetwarzaniem i pracą w

czasie rzeczywistym. Systemy 1080 i 1090 oparte są o nowe jednostki centralne KL-10 zbudowane na logice ECL/MSI z czasem dostępu do pamięci 15 ns. Lista rozkazów obejmuje 386 instrukcji. Pamięć operacyjna może mieć pojemność od 128 K do 4 milionów K słów 36-bitowych. (lj)

● Na dachu budynku fabryki firmy FAIRCHILD montującej obwody scalone usadowił się patrol 30 uzbrojonych Indian z plemienia Narajo. Zajęli oni fabrykę, żądając **przywrócenia do pracy 500 zwolnionych Indian** spośród 1000 zatrudnionych przez firmę, oraz polepszenia warunków socjalnych. Przedstawiciel firmy zagroził zamknięciem fabryki i przeniesieniem produkcji do innej miejscowości. Warto zaznaczyć, że tereny fabryczne to obszar rezerwatu, czyli własność Indian wydzierżawiana tylko firmie. MAT.

● **Suma obrotów zagranicznych sprzętem informatycznym** w styczniu br. w USA osiągnęła 210 mln. dol. Import wyniósł 17,9 mln. dol. a export 188,7 mln. dol. Blisko 50% exportu w styczniu dotyczyło urządzeń peryferyjnych, a 25% dotyczyło komputerów cyfrowych i analogowych. Jeżeli chodzi o export sprzętu do przetwarzania danych to spadł on w stosunku do roku 1974 z 165,2 mln. dol. do 33,3 mln. dol. MAT.

Obroty rynku minikomputerowego

Zachodnio-europejski rynek minikomputerowy zanotował w roku 1973 obroty sięgające 67 milionów funtów (angielskich). Według przewidywań specjalistów obroty osiągną sumę 448 milionów funtów w roku 1983. (lj)
Control and Instrumentation, December 1974

Mini-miero HEWLETT-PACKARD

Nowy model HEWLETT-PACKARD 55 zaanonsowany na początku tego roku, ma ten sam sposób programowania, co — znany również w Polsce — model HP 65, ale nie dysponuje pamięcią zewnętrzną i jest ograniczony do 50 kroków programu. Natomiast jego cena — co brzmi zachęcająco — jest o połowę niższa od ceny HP 65. Zwraca też uwagę jego podstawowe przeznaczenie: pomoc w programowaniu. Programista może na tym małym kalkulatorku sprawdzać poprawność optyczną pisanego właśnie programu w COBOLU bądź w PL/1, co redukuje do minimum niebezpieczeństwo błędów przy wczytywaniu programu do komputera. (elk.)

Komputer — detektywem

Komputer jako detektyw pracuje w systemie opracowanym przez firmę amerykańską CALSPAN Corp. System minikomputerowy wraz z radiołączem do dużej maszyny jest zainstalowany na mikrobusie.

Dane **na temat wypadku** zawierające opis pojazdów, śladów kół, poślizgów, rodzaje **uszkodzeń**, charakterystykę terenu itd. uzyskane od kierowców i świadków są przekazywane do głównej maszyny. Tam maszyna dokonuje analizy i na ekranie lub drukarce zostaje przedstawiona najbardziej prawdopodobna wersja wypadku. (lj)
Industrial Research, October 1974

Komputery i meteorologia

„Biuro meteorologiczne przyszłości" powstało ostatnio w Waszyngtonie. Eksperymentalny system, który kosztował 1 milion dolarów obejmuje minikompter z pamięcią 64 k oraz zespół monitorów ekranowych wyświetlających mapy, wykresy i inne dane na temat pogody, uzyskiwane w czasie około 1/40 czasu wymaganego w eksploatowanych dotąd systemach na drukowanie odpowiednich dokumentów. Planuje się uruchomienie 70 takich systemów w całym kraju do roku 1980. (lj)
Industrial Research, November 1974

Minikomputer R-5

Na Węgrzech pracuje się nad najmniejszym modelem maszyny cyfrowej Jednolitego Systemu o nazwie R-5. Według publikacji węgierskich ma to być „komputer do specjalnych zastosowań". Jednocześnie prowadzone są prace nad rodziną maszyn o nowej koncepcji. Działalność na tym polu przejawiają zakłady VIDEOTON (sprzęt) i INFELOR (oprogramowanie). (lj)
New Hungarian Exporter, November--December 74.

Planar News, 5 November 1974

Rywalizacja firm komputerowych

Tygodnik angielski "The Economist" z dnia 11 stycznia 1975 r. zamieszcza interesującą wiadomość na temat rywalizacji przedsiębiorstw komputerowych na terenie Wielkiej Brytnii.
Po raz co najmniej trzeci od czasu przejęcia w lutym ubiegłego roku władzy przez rząd Labourzystowski, minister przemysłu, Antony Benn, wtrącił się do interesów innego ministerstwa w celu uzyskania dla brytyjskiej firmy ICL zamówienia na sprzęt komputerowy do celów publicznych. Poprzednio nie zdążył on zapobiec zakupowi przez Ministerstwo Spraw Wewnętrznych dla policji Doliny Tamizy sprzętu od firmy HONEYWELL. Zakup od tej samej firmy komputera dla celów pośrednictwa pracy (w Ministerstwie Pracy), zdołał ograniczyć jedynie do jednorocznego próbnego zamówienia.

Obecnie próbuje on wstrzymać Ministerstwo d/s Środowiska Naturalnego przed wydaniem zgody Kolejom Brytyjskim na zakup w firmie UNIVAC systemu rezerwacji miejsc w pociągach pospiesznych za sumę 15 milionów funtów.

Amerykańska firma UNIVAC w sposób oczywisty wygrała otwarty przetarg, pokonując IBM. Te dwie firmy dominują w zakresie systemów rezerwacji, miejsc, natomiast HONEYWELL trzecia amerykańska firma z zaproszonych do przetargu przez Koleje Brytyjskie, odmówiła udziału. Propozycje brytyjskiej firmy ICL zostały uznane za nierealistyczne i odrzucone.

Firma ICL proponowała zastosowanie swojego **nowego i niedostatecznie** wypróbowanego komputera 2980, co do którego właściwego działania nie są jeszcze pewni nawet jego konstruktorzy. Oprócz tego ICL ma stosunkowo małe doświadczenie w realizacji systemów rezerwacji miejsc, co oznacza, że liczba projektantów systemów i programistów jaką dysponuje firma nie jest wystarczająca, aby zrealizować system w terminie wymaganym przez Koleje Brytyjskie tzn. do roku 1977.

Dla spełnienia tego warunku UNIVAC planuje instalację swojego wypróbowanego modelu komputera i rozpoczęcie pracy nad najważniejszymi programami jeszcze w połowie bieżącego roku.

ICL przedstawił również nowe propozycje dotyczące instalacji mniejszego i tańszego modelu nowej serii. Jednak i te propozycje zostały odrzucone w szczególności z tego powodu, że ostateczna cena systemu miała zostać określona dopiero po jego ukończeniu.

W międzyczasie centralna agencja komputerowa, która stoi w Wielkiej Brytanii na straży zakupów maszyn dla celów publicznych zawiadomiła ministra Benn'a, że „z punktu widzenia technicznego i ekonomicznego, Koleje Brytyjskie dokonały prawidłowej decyzji". (lj)

Problemy oprogramowania i zadania, jakie w związku z jego produkcją stoją przed Zjednoczeniem Informatyki

Publikowany poniżej artykuł jest, opracowaną na podstawie materiałów udostępnionych przez dyrekcję Zjednoczenia Informatyki, próbą określenia zadań programowych Zjednoczenia w dziedzinie oprogramowania użytkowego i narzędziowego. Wiąże się to z koncepcją przemysłowej produkcji oprogramowania, w której Zjednoczenie Informatyki upatruje swoje najważniejsze zadanie.

Udział oprogramowania w wartości systemów informatycznych wykazuje w ostatnich latach stałą tendencję wzrostu i w czołowych krajach świata znacznie przekracza obecnie udział sprzętu (hardware'u).

Polski przemysł informatyczny stoi przed pilną koniecznością intensywnego wzrostu produkcji oprogramowania. Stąd zakładamy szybki rozwój tej produkcji naszego Zjednoczenia. Całość oprogramowania produkowanego w przyszłości w sieci ZETO, podzielić można na oprogramowania użytkowe, oprogramowanie narzędziowe i oprogramowanie podstawowe.

Oprogramowanie użytkowe

Biorąc pod uwagę istniejące i przyszłe zapotrzebowanie oraz charakter działalności ośrodków zetowskich, jak również ich dorobek i doświadczenia, Zjednoczenie Informatyki przeznaczy większą część swego potencjału kadrowego na produkcję oprogramowania użytkowego. Chodzi tu o programy i systemy, które bezpośrednio obsługują odbiorcę w różnych obszarach i rodzajach działalności, takich jak ewidencja środków materialnych i gospodarka osobowa, zaopatrzenie, rachunek kosztów, planowanie, kontrola nakładów, informacja naukowa i techniczna, badania, prace projektowo-konstruktorskie itp.

ZI zmierza tu przede wszystkim do podniesienia efektywności procesu wytwórczego. Bardzo mocnym środkiem zwiększającym efektywność tego procesu jest p o w t a r z a l n o ś ć osiągniętych rozwiązań. Szerokie wprowadzenie powtarzalnego oprogramowania wielokrotnie obniża nakłady na realizację systemów komputerowych. Dlatego, niezależnie od własnej inwencji, zamierza się adaptować w jak największym stopniu oprogramowanie zagraniczne, nie wykluczając w uzasadnionych przypadkach zakupu najbardziej nowoczesnych rozwiązań na zasadach licencji i kooperacji.

Przewiduje się, że w roku 1980 w zespołach wytwarzających oprogramowanie użytkowe będzie pracowało około 2100 osób, to jest 70% ogółu zatrudnionych w produkcji oprogramowania. Zjednoczenie widzi więc w systemach użytkowych główny kierunek swej software'owej specjalizacji. Przewiduje się dalszy podział pracy pomiędzy przedsiębiorstwa i zespoły zetowskie według dziedzin zastosowań i rodzaju odbiorcy.

Docelowo Zjednoczenie Informatyki stawia przed tą grupą wyrobów następujące podstawowe wymagania:

- systemy oprogramowania użytkowego należy konstruować w ten sposób, aby czas od rozpoczęcia prac nad systemem u odbiorcy do uruchomienia eksploatacji nie przekraczał dla większości modułów okresu sześciu miesięcy
- oprogramowanie powinno stwarzać dogodne warunki dla zmian organizacyjnych u użytkownika i dla modyfikacji istniejącego systemu informatycznego
- przyswojenie typowego, powtarzalnego oprogramowania użytkowego powinno być na tyle łatwe, by odbiorca nie musiał w tym celu angażować własnych, wysokokwalifikowanych informatyków profesjonalnych.

Rozbudowując w tej dziedzinie potencjał wytwórczy, zakłady Zjednoczenia będą przechodziły stopniowo od stosowanych dotychczas pakietów użytkowych programów prefabrykowanych do bardziej nowoczesnych rozwiązań, powtarzalnego oprogramowania parametryzowanego.

Przewiduje się, że w produkcji oprogramowania użytkowego zastosowane zostaną na szeroką skalę generatory oprogramowania, wykorzystujące komputer jako silny instrument wspomagający czynności procesu projektowania i programowania użytkowego, zaliczamy również systemy poświęcone zastosowaniom do automatyzacji prac zawodowych. Większa część oprogramowania przygotowanego w Zjednoczeniu Informatyki dla maszyn Jednolitego Systemu, będzie miała także charakter oprogramowania użytkowego.

Biorąc pod uwagę możliwości, które otwierają się przed zastosowaniami techniki komputerowej w Polsce, ZI przyjmuje, iż jednym z najważniejszych obowiązków Zjednoczenia jest służyć pomocą gospodarce kraju w poprawie stanu zarządzania i organizacji. Większa część zadań, wykonywanych przez informatykę krajową w tej dziedzinie przypadnie na systemy obiektowe, obsługujące przedsiębiorstwa, kombinaty, wielkie organizacje gospodarcze, zjednoczenia przemysłowe, handlowe itp. Zjednoczenie zaangażuje około 30% swego potencjału wytwórczego oprogramowania w projektowanie i oprogramowanie systemów powtarzalnych. W latach 1974—80 łączna produkcja tego oprogramowania osiągnie wartość 3 tys. osobolat, a zespoły zetowskie pracujące nad tą problematyką będą liczyły w roku 1980 około 900 osób. Podstawową jednostką dla której Zjednoczenie zamierza opracowywać systemy powielarne będzie branża, zjednoczenie, WOG w odniesieniu do organizacji przemysłowych oraz odpowiadające im organizacje w innych działach gospodarki narodowej.

Zaangażowanie sił i środków dla tego typu jednostek pozwoli uzyskać niezbędny stopień kompleksowości rozwiązań systemów informatycznych w zakresie przetwarzania danych dla potrzeb zarządzania. Projektowane i oprogramowane dla potrzeb wymienionych wyżej organizacji systemy, składać się będą z powielarnych podsystemów, jednostek funkcjalnych i modułów. Elementy te będą spełniały wymogi określone dla poszczególnych grup standardowego oprogramowania użytkowego.

Wybór branż, WOG-ów, zjednoczeń, dla których tego rodzaju działanie ZI zostanie podjęte, będzie uwzględniał preferencje dla organizacji najbardziej istotnych z punktu widzenia interesu społecznego.
Kompleksowa obsługa danej branży, stanowić będzie podstawę dla specjalizacji konkretnych ośrodków obliczeniowych.

Ten model działania zabezpiecza użytkownikowi możliwości pełnej implementacji opracowanych systemów dla wszystkich jego jednostek organizacyjnych, bowiem przez ośrodek projektujący będzie on miał do dyspozycji moc obliczeniową całej organizacji ZETO. Odciążenie użytkownika od złożonej i skomplikowanej problematyki technologii przetwarzania, pozwoli mu skoncentrować się na zagadnieniach racjonalizacji własnej struktury organizacyjnej, społecznej, poprawie metod działania itp. Zjednoczenie Informatyki zamierza współpracować przy realizacji tego modelu z innymi resortowymi, branżowymi i zakładowymi ośrodkami informatycznymi, posiadającymi sukcesy w dziedzinie wdrażania i eksploatacji systemów.

Przewidywane zasoby oprogramowania powtarzalnego pozwolą w połowie pięciolatki 1976—80, przyjąć zasadę powszechnego stosowania oprogramowania powielarnego w zakresie systemów obiektywnych.

Zjednoczenie Informatyki będzie oferowało typowe prefabrykowane systemy automatyzacji systemów obiektywnych (ASO): (a) instytucjom nie posiadającym własnej maszyny lub zaawansowanej kadry informatycznej, (b) odbiorcom dysponującym własnym sprzętem i własną kwalifikowaną ka-

drą, co umożliwi tej grupie odbiorców podjęcie prac nad systemami, złożonymi ze współdziałających ze sobą elementów prefabrykowanych i własnych.
W oparciu o te przesłanki, przy założeniu wystarczająco dużych dostaw maszyn, Zjednoczenie obejmie w ciągu 7—10 lat różnymi formami swej działalności w zakresie zastosowań informatyki do zarządzania i organizacji 50—70% obiektów gospodarczych w kraju, z preferencją dla ugrupowań najbardziej istotnych z punktu widzenia interesu społecznego.

Proponowana linia rozwoju umożliwi zaoszczędzenie znacznych sił i środków po stronie wytwórców oprogramowania, jak również po stronie użytkowników maszyn cyfrowych oraz ułatwi wygospodarowanie potencjału informatycznego niezbędnego w innych, bardzo efektywnych zastosowaniach do organizacji i kierowania. Takim kierunkiem zastosowań jest sterowanie procesami technologicznymi. Przewiduje się coraz szybszy wzrost nakładów na ten kierunek i komutowanie systemów ASO z systemami automatyzacji procesów technologicznych (APT) lub pokrewnymi. ZI wydzieli dla tej problematyki około 10% kadry pracującej nad systemami ASO.
Do oprogramowania związanego z automatyzacją prac zawodowych (APZ) zalicza się m.in. translatory bardzo prostych, łatwych do opanowania języków maszynowych, specjalizowanych według zainteresowań zawodowych.
Zapotrzebowanie na systemy APZ wzrasta w miarę instalowania maszyn z wielodostępem i wyższej klasy mikrokomputerów, które szybko zwiększają krąg użytkowników maszyn cyfrowych, dostarczając wyniki przewarzania niemal natychmiast po sformułowaniu problemu.
Efektywność systemów APZ zasługuje na szczególne podkreślenie. Dotyczy to zwłaszcza prac projektowo-konstruktorskich. Maszyna wielokrotnie skraca czas potrzebny na opracowanie nowego wyrobu lub nowej technologii. W niektórych dziedzinach, kiedy w grę wchodzi wysoce złożona struktura wewnętrzna produktu, konkurencyjność nie może być już uzyskana bez udziału komputera w procesie innowacji. Produkcja wielodostępowego i mikrokomputerowego oprogramowania zawodowego jest bardzo korzystna również z uwagi na to, że zagadnienia, które się rozwiązuje za pomocą tego oprogramowania można łatwiej opisywać i algorytmizować, niż to ma miejsce w zastosowaniach do organizacji i kierowania.

Te wszystkie przyczyny powodują, że Zjednoczenie Informatyki traktuje produkcję i rozpowszechnianie wielodostępnego i mikrokomputerowego oprogramowania prac zawodowych jako nadzwyczaj ważną dziedzinę swej działalności. Czynnikiem limitującym jest tutaj bariera sprzętu. Licząc się z realiami, program ZI przewiduje, że w latach 1979—80 20—25% naszego potencjału wytwórczego w dziedzinie oprogramowania skupi się w zespołach pracujących nad systemami APZ.

Zespoły zetowskie, pracujące nad problemami APZ, osiągną w roku 1980 liczebność 600—750 osób. Można więc przyjąć, że zespoły te będą mogły w pięcioleciu 1976—80 podjąć się roli wspomagającej w stosunku do większości instytucji, które są obecnie potencjalnymi użytkownikami systemów APZ.

Zjednoczenie Informatyki upatruje w koncepcji Jednolitego Systemu EMC jedną z głównych wytycznych dla swego rozwoju. Zakłada się, że Zjednoczenie wniesie znaczny wkład w produkcję oprogramowania dla maszyn JS.

Przyjmując, że w latach 1974—80 tylko połowa oprogramowania, wytworzonego w jednostkach ZI, osiągnie niezbędne standardy, produkcja Zjednoczenia w tej dziedzinie rozpowszechniana w krajach RWPG osiągnie do roku 1980 łączną wartość 5000 osobolat. Koncepcja Jednolitego Systemu oznacza, że oprogramowanie to wejdzie do biblioteki programów zasilających pracę wielu tysięcy komputerów, instalowanych w krajach obozu socjalistycznego. Wynika stąd, że eksport Zjednoczenia, liczony w wartości udostępnionego oprogramowania oraz w liczbie, dla których jest ono przeznaczone, znacznie przekroczy kwoty, realizowane obecnie przez duże firmy komputerowe Europy Zachodniej.
Realizacji tych zamierzeń służy przyjęte w programie ZI ukierunkowanie produkcji na powtarzalne oprogramowanie warsztatowe i podstawowe, pozwalające: (a) zwiększać efektywność procesu projektowo-programistycznego (b) oszczędzać prace zespołów projektantów i programistów, (c) podnosić sprawność, a także dostępność systemów komputerowych.
Prace nad oprogramowaniem komputerowym Jednolitego Systemu są już prowadzone w ZI w ramach tematu „Awtomatizirowannyje Sistiemy Uprawlenija" (ASU) w Ośrodku Badawczo-Rozwojowym Informatyki. Oprogramowanie JS jest też przedmiotem działalności Zkładu R-POL-SYSTEM w przedsiębiorstwie ZOWAR.

Oprogramowanie narzędziowe

Więcej niż 90% programów użytkowych, pracujących obecnie w kraju powstaje w niewielkich zespołach tworzonych przeważnie przez samego użytkownika. Produkcja tego oprogramowania ma charakter rzemieślniczy. Powtarzalność oprogramowania użytkowego, która w programie ZI jest jednym z głównych środków podniesienia efektywności i uprzemysłowienia procesu projektowania i programowania, nie jest środkiem wystarczającym. Konieczne jest odpowiednie wyposażenie projektantów i programistów w zawodowe narzędzia pracy. Zjednoczenie, jako największa w kraju instytucja profesjonalnej informatyki stawia przed sobą zadanie takiego uzbrojenia swych zespołów projektowo-programistycznych, by ich praca nabrała cech nowoczesnego procesu przemysłowego. Należy pamiętać przy tym o istotnych elementach jakie charakteryzują produkcję oprogramowania i stanowią bardzo ważną jej specyfikę, odmienną od specyfiki sprzętowej. Wśród narzędzi stosowanych w projektowaniu systemów i programowaniu coraz większą rolę odgrywa sam komputer oraz oprogramowanie pełniące funkcję środków produkcji w przemyśle software'owym. Zakładamy znaczne zaangażowanie potencjału ZI w produkcję oprogramowania warsztatowego i narzędziowego. Wymienić tu należy przede wszystkim programy obsługujące zbiory danych i ułatwiające programowanie określonych typowych przebiegów maszyny, prekompajlery, programy wspomagające uruchamianie dużych programów, flowchartery itp. Tego rodzaju oprogramowanie inżynierii oprogramowania wielokrotnie zwiększa wydajność pracy. Należy przypuszczać, że jeden z rozwijanych obecnie nowych kierunków techniki programowania, łączący programy narzędziowe ze standardowymi blokami parametryzowanego oprogramowania użytkowego, pozwoli oprzeć znaczną część produkcji oprogramowania w ZI jeszcze przed rokiem 1980 na wspomnianych wyżej komputerowych generatorach oprogramowania.

Oprogramowanie podstawowe

W kolejnych generacjach maszyn cyfrowych zwiększa się rola oprogramowania podstawowego, na który składają się programy sterujące maszyny i zarządzające wykonaniem prac (system operacyjny), programy kontrolno-diagnostyczne, programy koordynacyjne wielodostępu i teletransmisji, jak również programy zapewniające współdziałanie systemów wielokomputerowych. Rozwój tego oprogramowania zmierza do zwiększenia wydajności komputera, skrócenia czasu odpowiedzi w konwersacji człowiek-maszyna, zmniejszenia strat wynikających z nie dość efektywnej wewnętrznej organizacji systemów cyfrowych oraz do stworzenia bardziej elastycznych i szerzej wykorzystywanych systemów informatycznych.

Do oprogramowania podstawowego zalicza się również translatory języków programowania różnego rzędu.

Oprogramowanie podstawowe, jako ściśle związane z modelem, budową i konfiguracją maszyny, jest opracowywane przeważnie przez producenta sprzętu. W specyficznej sytuacji najbliższych lat Zjednoczenie Informatyki widzi uzasadnienie dla rozwoju swej działalności również w tym kierunku. Będzie to aktywność uzupełniająca. Głównym celem Zjednoczenia w produkcji oprogramowania podstawowego jest tworzenie przesłanek dla lepszego zaspokojenia potrzeb użytkownika i rozszerzenie możliwości współpracy i dialogu z producentem sprzętu.

Wyniki II Krajowego Konkursu na Prace Naukowe, Popularnonaukowe, Doktorskie i Magisterskie w Dziedzinie Informatyki

3 marca 1975 r. Minister Nauki, Szkolnictwa Wyższego i Techniki — prof. dr hab. Sylwester Kaliski — wręczył nagrody laureatom II Krajowego Konkursu na prace naukowe, popularno-naukowe, doktorskie i magisterskie w dziedzinie informatyki.

Do konkursu zakwalifikowano 43 prace (po wstępnej selekcji zwrócono autorom prace, które nie dotyczyły informatyki). Każdą z zakwalifikowanych prac szczegółowo opiniowali naukowcy i specjaliści, zgodnie z wymogami prawa autorskiego i według jednolitych kryteriów ocen.
Za kryteria oceny przyjęto:

- stopień, w jakim praca dotyczy informatyki
- stopień trudności tematu
- w jakim stopniu praca jest nowatorska
- oryginalność ujęcia tematu
- znaczenie pracy dla rozwoju informatyki
- znaczenie pracy dla popularyzacji informatyki
- czy praca prezentuje temat w sposób wyczerpujący
- zalety dydaktyczne
- kryterium własne oceniającego.

Zastosowano następującą punktację: ocena b. wysoka — 4 punkty, wysoka — 3 punkty, średnia — 2 punkty, niska — 1 punkt, b. niska — 0 punktów.
Przewodniczącym Sądu Konkursowego został powołany przez Ministra Nauki, Szkol-

nictwa Wyższego i Techniki, prof. dr hab. Leon Łukaszewicz. W skład sądu weszli wybitni naukowcy — przedstawiciele środowisk naukowych, Komitetu Informatyki PAN, Polskiego Komitetu Automatycznego Przetwarzania Informacji NOT, Sekcji Maszyn i Systemów Cyfrowych SEP-NOT, Państwowego Wydawnictwa Naukowego. Sąd Konkursowy przedstawił prace do nagród oraz wysunął wnioski zmierzające do zmiany regulaminów następnych Konkursów.

W grupie prac naukowych nagrody otrzymali:

II Nagrodę:

Krystyna JERZYKIEWICZ, Jerzy SZCZEPKOWICZ — za pracę „ALGOL 1204 — System programowania maszyny cyfrowej ODRA 1204". PWN, 1973 (wznowienie).

Pozycja ta zasługuje na szczególną uwagę ze względu na duże możliwości praktycznego wykorzystania. Sama praca stanowi bardzo dobry podręcznik użytkowania komputera ODRA 1204, programującego w ALGOLU 1204. W swoim czasie stworzyła ona możliwość oprogramowania i wykorzystania komputera ODRA 1204.

II Nagrodę:

Dr inż. Jan ŻYDOWO — za pracę „Zastosowanie metody symulacji cyfrowej do planowania produkcji jednostkowej" Wydawnictwa OBRI INFORNA, 1973.

W warunkach polskich praca ma charakter pionierski. Stanowi pomost pomiędzy rozwijającą się u nas dość szybko teorią metod symulacyjnych, a nad wyraz jeszcze skromną praktyką. Opracowanie oryginalne, znalazło praktyczne zastosowanie.

WYRÓŻNIENIE:

dr med. Andrzej BRODZIAK — za pracę „Struktury i procesy informacyjne centralnego systemu nerwowego człowieka. Teoria i jej pierwsze praktyczne wykorzystanie". Nakładem MNTSW i PAN, seria INFORMATYKA MEDYCZNA, 1973 r.

Praca z zakresu informatyki biomedycznej przedstawiająca teorię procesów informatycznych w nawiązaniu do struktury i czynności ośrodkowego układu nerwowego człowieka. Wyniki dokonanej analizy struktury i czynności procesów informacyjnych makroustroju biologicznego posłużyły Autorowi do opracowania podstaw teoretycznych Szpitalnego Systemu Informacyjno-Nadzorującego Katowice — 4 (SSINKA-4).

W grupie prac doktorskich nagrodę otrzymał:

III Nagrodę

Dr Jan Małuszyński — za pracę „O pewnej metodzie rozpoznawania języków bezkontekstowych" PWN, 1972, prace naukowe COPAN.

Praca zajmuje się jednym z ważnych problemów, występujących przy konstrukcji translatorów języków programowania. Efektywność rozpoznawania ma bowiem istotny wpływ na efektywność tłumaczenia programów. Praca jest nowatorska i stanowi istotny wkład do teorii programowania.

W grupie prac magisterskich nagrody otrzymali:

II Nagrodę

Mgr Stanisław KOWALIK — za pracę „Projekt pamięci buforowej", napisaną na Politechnice Gdańskiej, Wydział Elektroniki, Instytut Informatyki pod kierunkiem doc. dr inż. Tadeusza Bartkowskiego w 1973 r.

Praca stanowi podstawę do opracowania szczegółowego projektu technicznego szybkiej półprzewodnikowej pamięci buforowej pośredniczącej między stosunkowo wolną główną pamięcią ferrytową komputera, a procesami komputera. Wyczerpujące omówienie tematu, duża użyteczność praktyczna.

Wyróżnienie

mgr Leszek MACIASZEK — za pracę „Rozliczanie płac agentów stacji benzynowych przy zastosowaniu ETO w Wojewódzkim Przedsiębiorstwie Obrotu Produktami CPN" napisaną w Wyższej Szkole Ekonomicznej we Wrocławiu — Wydział Gospodarki Narodowej, Instytut Metod Rachunku Ekonomicznego, Zakład Informatyki pod kierunkiem mgr Andrzeja Ramułta — zakończona w 1972 r.

W przypadku CPN problem rozliczeń placowych jest dostatecznie skomplikowany, aby uzasadnić zastosowanie maszyn cyfrowych. Praca może być traktowana jako projekt wstępny. Charakteryzuje się systematycznym ujęciem oraz logicznym rozwinięciem problemu.

Wręczając nagrody i składając laureatom gratulacje Minister Nauki, Szkolnictwa Wyższego i Techniki podkreślił, że w przyszłości będą nagradzane przede wszystkim prace nadające się do praktycznego wykorzystania.

Wnioski Sądu Konkursowego

- należy spopularyzować dalsze konkursy Ministra Nauki, Szkolnictwa Wyższego i Techniki w dziedzinie informatyki w gronie pracowników nauki na wyższych uczelniach. Celowe byłoby rozważenie możliwości spopularyzowania konkursu przy współudziale Sekretarza Naukowego PAN i Sekretarza Generalnego NOT, lub organizowanie konkursów przy ich współudziale
- proponuje się powołanie Prezydium Sądu Konkursowego znacznie wcześniej tzn. po ogłoszeniu konkursu, aby mogło ono uczestniczyć w toku prac przygotowawczych
- do warunków Konkursu należy wprowadzić obowiązek uzasadnienia zgłoszenia pracy przez kierownika placówki zgłaszającej
- oprócz istniejącego dotychczas podziału należy zastosować dalszy podział na prace z zakresu: sprzętu, oprogramowania, zastosowań
- nagrodzone mogą być prace, które spełnią warunek „solidna praca o znaczeniu praktycznym"
- ogłosić III Konkurs na prace z zakresu informatyki.

K. Hajduk-Popławska
Sekretarz Konkursu

Komputery na Opolszczyźnie

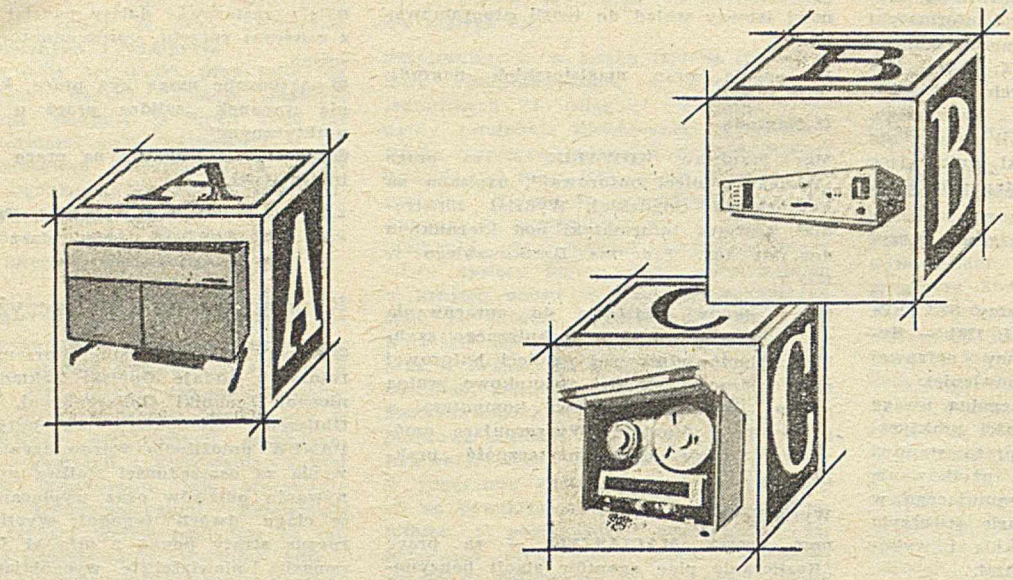
- Coraz większe usługi przemysłowi i rolnictwu oddaje Opolski Zakład Elektronicznej Techniki Obliczeniowej.

Obliczenie komputera dla huty MAŁAPANEW umożliwiło wprowadzenie premiowania za oszczędności staliwa przy wykonywaniu odlewów oraz wypłacanie premii w ciągu dwóch tygodni; wyeliminowano roczne straty ponad 2 mln zł jakie huta ponosiła niewłaściwie wyceniając odlewy. Bieżące rozliczanie — przy pomocy komputera — zużycia skór importowanych w kombinacie OTMĘT zwiększyło oszczędności o ok. 2 proc. uzyskano 15 tys. m kw. skór wartości 2 mln zł. ETO pozwoliła ujawnić wiele nieprawidłowości w Fabryce Obrabiarek w Kuźni Raciborskiej, ułatwiając właściwe rozliczanie się z części maszyn. Jeszcze tej wiosny komputery pomogą opolskim rolnikom — przekazując dane wskazujące zakres potrzeb biologicznych poszczególnych roślin w podstawowych składnikach mineralnych — z uwzględnieniem rodzaju gleby oraz warunków klimatycznych. (kr)

- Na razie siedmiu raciborskich informatyków dostało od tamtejszego Urzędu Miejskiego dwa pokoiki, w których przygotowują się do świadczenia usług informatycznych raciborskim fabrykom i rolnictwu.

Według planów, pod koniec bieżącego roku raciborski oddział ZETO Opole otrzyma specjalnie zaadaptowany gmach, gdzie znajdzie zatrudnienie 160 osób i zainstaluje się maszyny ODRA i RIAD. Z usług ośrodka korzystać będą: Raciborska Fabryka Kotłów RAFAKO (ukończono już prace nad projektem wstępnym ogólnozakładowego planowania produkcji), ZELMER, RAFAMET w Kuźni Raciborskiej.

Zupełną nowością będzie zastosowanie komputera do rozwiązania problemu ekonomicznego stosowania nawozów w rolnictwie. Doświadczalnie, już jesienią tego roku, komputer będzie porcjował nawozy na polach kombinatu PGR w Głubczycach. Na wyniki tego eksperymentu czeka całe rolnictwo. (elk)



Programowanie – opis danych w języku programowania (grupa 07.04)

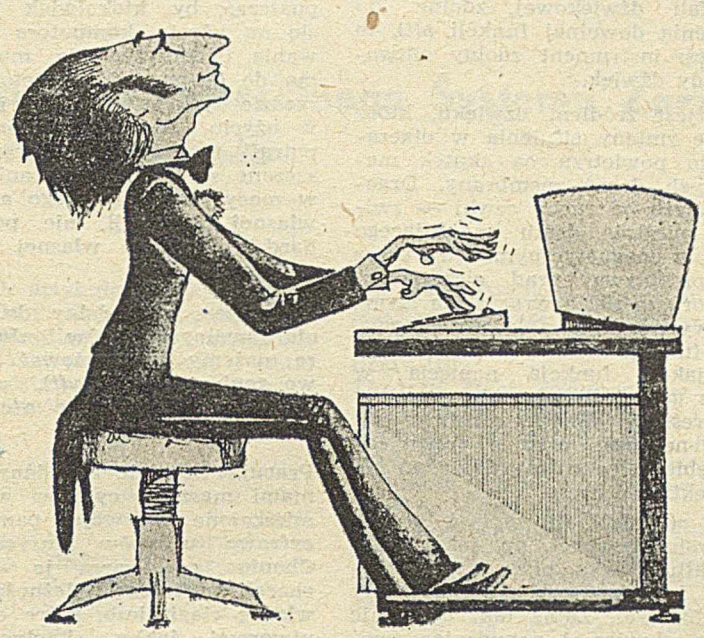
- **identyfikator** (ang. *indentifier*) — znak lub grupa znaków użytych do identyfikacji lub nazwania określonej pozycji danych oraz w miarę możliwości również do wskazania cech tych danych.
- **nazwa danej** (ang. *data name*) — znak lub grupa znaków użytych wyłącznie do identyfikacji określonej pozycji danych.
- **stała symboliczna** (ang. *figurative constant*) — nazwa danej, zarezerwowana dla określonej stałej w określonym języku programowania.
- **nazwa określająca** (ang. *qualified name*) — nazwa danej z wyraźnym sprecyzowaniem klasy, do której dana ta należy w określonym systemie klasyfikacyjnym.
- **subskrypt** (ang. *subscript*) — symbol związany z nazwą zbioru i służący do identyfikacji określonego podzbioru lub elementu.
- **indeks** (ang. *index*) — subskrypt, który identyfikuje lokalizację określonej pozycji danych z uwzględnieniem niektórych innych pozycji danych.
- **numer poziomu** (ang. *level number, rank*) — liczba odniesienia, wskazująca lokalizację pozycji w układzie hierarchicznym.
- **słowo zarezerwowane** (ang. *reserved word*) — słowo w języku źródłowym, którego znaczenie jest ustalone przez określone reguły tego języka i nie może być zmienione w celu udogodnienia jakiegokolwiek programu napisanego w tym języku źródłowym. Reguły języka źródłowego mogą również zabraniać użycia tych słów w innych kontekstach niż ustalone.
  Przykłady:
  1. „SIN" może być słowem zarezerwowanym dla wywoływania podprogramu obliczania funkcji sinus.
  2. W języku COBOL t.zw. „słowa COBOLOWE".
- **obraz** (ang. *picture*) — opis sznura znaków w języku programowania, w którym każda pozycja związana jest z symbolem reprezentującym cechy znaku zawartego w tej pozycji.
  Przykład:
  W języku COBOL „9999" jest używane jako obraz każdego czterocyfrowego słowa numerycznego.
- **literał** (ang. *literal*) — pełne przedstawienie wartości pozycji w programie źródłowym, która nie może być zmieniona podczas translacji tego programu.
  Przykład:
  Słowo „FAIL" w instrukcji „Jeśli X = 0 drukuj FAIL".
- **wypełniacz** (ang. *filler*) — jeden lub więcej znaków związanych z określoną pozycją danych, które służą do doprowadzenia jej postaci do z góry określonych rozmiarów.
- **wypełnianie** (ang. *padding*) — dołączanie wypełniaczy do danych.

Władysław Klepacz

ERRATA do nr 5/75

1. w tytule haseł grupy 07.03: jest ... **transakcji** powinno być **translacji**
2. definicja hasła **asembler:** jest ... **języku programowym** powinno być **języku problemowym**
3. w definicji hasła **symulator** jest ... **programy przygotowania** powinno być **przygotowane**
4. w haśle **język symboliczny** oraz w definicji hasła **asemblować** wyrażenie **asemblować** powinno mieć tylko jedno s.

Komputery w muzyce



Opisywane miesiąc temu komputery wykonywały prace pomocnicze, jak gdyby na tyłach muzyki — pora omówić ich akcję na pierwszej linii frontu, czyli udział w bezpośrednim tworzeniu muzyki. Dadzą się tu wyodrębnić trzy główne kierunki natarcia. A więc w kolejności, zaczynając od najmniej złożonej operacji: sterowanie elektronicznym sprzętem muzycznym, syntezę dźwięków i kompozycję utworów muzycznych. Podział taki jest oczywiście umowny, granice między poszczególnymi kierunkami trudne czasem do ustalenia. Przyjmijmy je tylko dla porządku i ustalenia uwagi. W dzisiejszym numerze omówimy zastosowanie maszyn do sterowania sprzętem muzycznym i część przesłanek teoretycznych umożliwiających zastosowanie ich do syntezy dźwięków.

Sterowanie przy użyciu komputera, zagadnienie ogromnej wagi, jest przedmiotem rozważań licznej kadry fachowców i ma już bogatą literaturę. Nie muzyka, rzecz jasna, znajduje się w centrum zainteresowania.

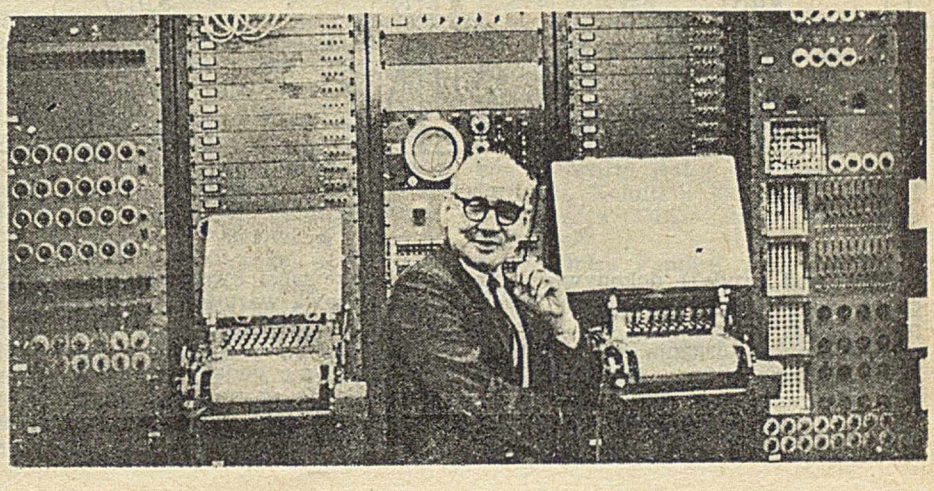
Tę samą niemal rolę będzie jednak spełniał komputer, gdy obrabiarki ze stosowanego komputerowo procesu technologicznego zastąpimy przez instrumenty muzyczne, które mogą być sterowane w sposób elektroniczny, jak na przykład instrumenty ze studia muzyki elektronowej. Bez pomocy komputera praca w takim studio jest nad wyraz uciążliwa i długotrwała. Trzeba ustawić pożądane parametry na każdym z urządzeń, sprawdzać jakość dźwięku, nagrywać go na taśmę, splatać z sobą nagrane osobno efekty — te czynności, wykonywane jedna po drugiej są ogromnie absorbujące. Na dowód tego przytoczmy fakt, że opracowanie napisanego z okazji XX Igrzysk Olimpijskich utworu Pendereckiego „Ekecherija" trwało w Studio Eksperymentalnym Polskiego Radia kilkaset godzin. Dodatkowe trudności powstają wówczas, kiedy dokonujemy zmian. Trudno wtedy trafić na poprzedni układ, bo trzeba zapamiętać położenie wszystkich pokręteł i przełączników pomiarowych.

Komputer ustawia je wszystkie naraz, z szybkością wielokrotnie większą niż szybkość ludzkiej ręki. Pozwala nie tylko tworzyć na bieżąco i jednocześnie je z sobą mieszać. Pamięta także sposób ich wytworzenia, czyli każde z kolejnych nastawień instrumentów. Nie musimy też operować poszczególnymi urządzeniami, wystarczy przekazać właściwy program maszynie cyfrowej. Mając do dyspozycji bibliotekę takich programów możemy modyfikować wcześniej nagrane utwory i zaznajamiać się z techniką innych kompozytorów.

Dzięki komputerowi niepomiernie wzrosła efektywność elektronicznego studia i zmienił się sposób jego wykorzystania.

Jeżeli usłyszymy dźwięk niezgodny z zamierzeniami — możemy natychmiast, podobnie jak dyrygent w orkiestrze, zatrzymać z pomocą komputera instrumenty i zmienić program szukając właściwego brzmienia. Co prawda, minikomputer kosztuje obecnie prawie tyle, co wszystkie urządzenia studia elektronicznego, ale rezultaty są nieporównywalne — za podwójną cenę uzyskujemy kilkanaście lub kilkadziesiąt razy lepsze wy-

Pionier muzyki komputerowej Milton Babitt w swoim studio



niki. Toteż studia sterowane komputerowo nie należą dziś do rzadkości. Najbardziej znane spośród nich to: studio eksperymentalne Bell Laboratories w New Jersey kierowane przez Maxa Matthewsa i Dicka Moorea, Columbia-Pincerton Electronic Music Center studio Petera Zinovieffa w Londynie wykorzystujące bardzo nowoczesny system MUSYS i oparte na podobnym systemie Electronic Music Studio zaprojektowane przez Knuta Wiggena w Sztokholmie oraz studio elektroniczne G. M. Köoniga na uniwersytecie w Utrechcie.

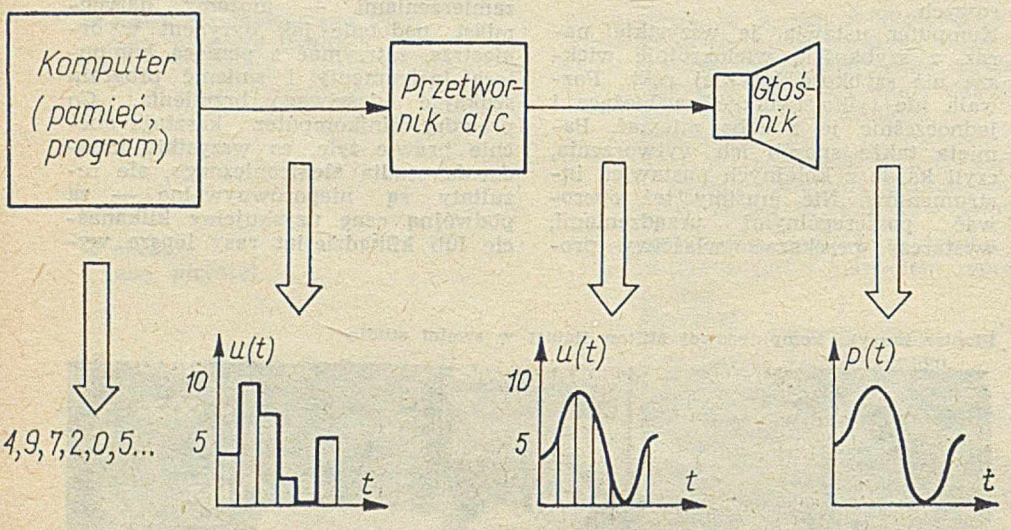
* * *

Drugim spośród głównych zastosowań komputerów w muzyce jest użycie ich do syntezy dźwięku. Zagadnienie to jest identyczne z celem, dla którego skonstruowano opisane w poprzednim numerze elektroniczne syntezatory dźwięku. Wielu zresztą muzyków uważa moogi za wszechstronne mózgi elektronowe i utożsamia je z maszynami cyfrowymi. Istotnie — elektroniczna struktura i poziom złożoności zbliża je do komputerów. Maszyny cyfrowe są rzecz jasna, bardziej doskonalsze, dysponują pamięcią mogą przetwarzać i zbierać dane, podejmując na ich podstawie decyzje. Na polu syntezy dźwięków moogi jednak niewiele im ustępują.

Aby szerzej wyjaśnić zasady syntezy dźwięku wypada przypomnieć parę znanych wszystkim pojęć fizycznych. Encyklopedyczna definicja dźwięku brzmi: „jest to zaburzenie falowe ośrodka materialnego, słyszalne dla ucha ludzkiego" itd. Dla naszych potrzeb to jedno zdanie wystarczy, by traktować dźwięk jako zmiany ciśnienia powietrza (no bo kto słucha kończnienie od czasu — $p(t)$. Wszystkie dźwięki mają swoje funkcje ciśnienia i każdy dźwięk można wywołać powodując drgania powietrza zgodnie z odpowiadającą temu dźwiękowi funkcją $p(t)$. Jeżeli zatem opracujemy źródło fali dźwiękowej, zdolne do wytworzenia dowolnej funkcji $p(t)$, to zbudujemy instrument zdolny odtworzyć każdy dźwięk.

Głośnik jest źródłem dźwięku, które wywołuje zmiany ciśnienia w otaczającym go powietrzu na skutek mechanicznych drgań membrany. Drgania te zależą od ruchu cewki — zwiniętego spiralnie drutu znajdującego się w polu magnetycznym. Gdy przez cewkę przepływa prąd elektryczny, cewka porusza się, a wraz z nią membrana, wytwarzając falę dźwiękową. Każdej funkcji ciśnienia $p(t)$ odpowiada jakaś funkcja napięcia w w cewce $u(t)$. Jeśli więc chcemy uzyskać określony dźwięk, musimy tylko wygenerować prąd o odpowiednim przebiegu, co należy do zadań typowo elektronicznych.

Funkcja $u(t)$ daje się zapisać w postaci czysto fizycznej, np. jako zmiany kształtu rowka płyty gramofonowej lub jako namagnesowanie taśmy magnetofonowej. Zapis taki odczytuje igła albo głowica, i pobudzając cewkę głośnika, powoduje powstanie fali dźwiękowej. Zapis funkcji $u(t)$ nie musi być dokonywany na płycie czy taśmie magnetofonowej, równie dobrze daje się przechowywać w pamięci maszyny cyfrowej. Będzie to naturalnie — jak wskazuje nazwa maszyny — zapis cyfrowy nie analogowy. Ale nie nastręcza to żadnych problemów: nie ma bowiem funkcji $u(t)$, której nie udaje



certów pod wodą?) dokonywane z częstotliwością od 20 do 16.000 drgań na sekundę (od 20 Hz do 16 kHz) wprawiające w drgania błonę bębenkową naszego ucha. Stałe ciśnienie jest odbierane jako cisza.

Zachowanie się dźwięku w czasie możemy wyrazić funkcją uzależniającą się próbkować i kwantować, tj. badać, jaką cyfrową wartość przyjmuje funkcja $n(t)$ co pewien czas, np. co jedną sekundę. Przy odczycie tej funkcji trzeba jedynie dodać urządzenie zmieniające zapis w postaci cyfr na zapis analogowy, tzw. przetwornik cyfrowo-analogowy i oto mamy już urządzenie zdolne do sterowania prądem dla głośnika niczym się nie różniące od popularnych środków zapisu dźwięku. Nie ulega jednak wątpliwości, że gdybyśmy się do tego ograniczyli, uzyskalibyśmy system o bardzo problematycznej przydatności. Trudno przypuszczać, by ktokolwiek zdecydował się na użycie komputera do zapisywania i odczytywania muzyki gdy ma do dyspozycji magnetofon — narzędzie tysiąckroć tańsze i łatwiejsze w użyciu. Ale magnetofon, chociaż potrafi precyzyjnie odtworzyć umieszczone na taśmie nagranie, nie jest w mocy dodać do niego ani szczypty własnej inwencji, nie potrafi tym bardziej tworzyć własnej magnetofonowej muzyki.

Komputer jest zdolny do generowania dowolnych ciągów liczbowych, które możemy interpretować jako cyfrowe zapisy funkcji $u(t)$, a więc po przejściu przetwornika a/c i głośnika jako dźwięki (rys. 1).

Pracując zgodnie z zadanymi programami maszyna cyfrowa analizuje umieszczone w swojej pamięci zapisy cyfrowe utworów muzycznych, np. Chopina, przetwarza je wychwytując charakterystyczne zależności i buduje własne ciągi liczb, które odpowiadają utworowi jakby chopinopodobnemu, ale różniącemu się od oryginałów. Jest to własne dzieło maszyny utrzymane w stylu znanych jej kompozycji.

Powyższy schemat jest oczywiście maksymalnie uproszczony. W dokładniejszych rozważaniach należałoby wziąć pod uwagę błędy wynikające z aproksymacji funkcji, przetwarzania analogowo-cyfrowego, zniekształcenia wprowadzane przez głośnik i szereg innych elementów. Trzeba by także uwzględnić w rys. 1 dodatkowe bloki elektroniczne zwiększające czytelność, prądu, który dostarczany jest do głośnika (wzmacniacze, filtry, itp.). Są to zagadnienia niezmiernie istotne — wystarczy wspomnieć, że dla zapewnienia dobrej jakości dźwięku należy podawać co najmniej 20 tysięcy cyfrowych wartości funkcji $u(t)$ na sekundę, czyli próbkować tę funkcję z częstotliwością 20 kHz. Pominiemy jednak te problemy, ponieważ chodzi nam o idee komputerowej syntezy dźwięku, a nie o jej techniczną realizację. Trudności i bez tego jest sporo. Najpoważniejszą z nich stanowi ogromna ilość danych, koniecznych do zapisania nawet krótkich utworów muzycznych. Zdarza się, że przekracza ona pojemności pamięci nawet dużych maszyn cyfrowych, a wymaga niezwykle szybkich i sprawnych programów. Aby stworzyć takie programy, trzeba opracować uniwersalny i skuteczny język zdolny do ujęcia skomplikowanych zestawów dźwięków.

Prace nad syntezą hamuje też fakt, że zbyt mało jeszcze wiemy o zależnościach między funkcją ciśnienia $p(t)$ i samym dźwiękiem. Prowadzone od wielu lat eksperymenty dostarczają jednak coraz dokładniejszych danych, co pozwala przypuszczać, że trudności te niezadługo będą usunięte.

Marek Hołyński

Koncepcja Uniwersalnego Systemu Zarządzania Bazą Danych dla komputerów JS

W kolejnych artykułach cyklu „Problematyka Bazy Danych" publikowanych w ostatnich ośmiu numerach INFORMATYKI przedstawiony został przegląd podstawowych problemów z zakresu systemów zarządzania bazami danych.

Ten i następne cztery kolejne artykuły poświęcone są omówieniu najistotniejszych elementów koncepcji Uniwersalnego Systemu Zarządzania Bazą Danych (USZBD) realizowanego przez Zakład Produkcji Oprogramowania R-POLSYSTEM na komputery Jednolitego Systemu.
Cel prezentacji koncepcji systemu jest wieloraki.

Przede wszystkim, chcemy poinformować Czytelników, iż R-POLSYSTEM rozpoczął prace mające na celu stworzenie uniwersalnego systemu zarządzania bazą danych na komputery JS. Sądzimy, że informacja ta pozwoli uniknąć dublowania prac, jak też pozwoli potencjalnym użytkownikom uwzględnić opracowywany przez ZPO R-POLSYSTEM USZBD w koncepcjach swoich systemów informatycznych.

Realizacja USZBD jest przedsięwzięciem poważnym i kosztownym. Mając na uwadze fakt, że realizowany przez ZPO — R-POLSYSTEM USZBD będzie zastosowany w wielu systemach informatycznych — popełnione w realizacji tego systemu błędy mogą się okazać szczególnie kosztowne.
Chcąc zmniejszyć do minimum niebezpieczeństwo popełnienia błędów w samej koncepcji, liczymy na krytyczne uwagi Czytelników zajmujących się problematyką baz danych, jak i potencjalnych użytkowników.

Nie bylibyśmy uczciwi, gdybyśmy pominęli wreszcie milczeniem fakt, że prowadzona przez nas na łamach INFORMATYKI prezentacja USZBD mieści się również w naszej działalności marketingowej.

Analizując dotychczasowe doświadczenia z zakresu oprogramowania obsługi baz danych dochodzimy do wniosku, że spośród całej mozaiki różnych rozwiązań możemy wyróżnić trzy zasadnicze ich klasy:

- SZBD, reprezentantem których jest koncepcja Komitetu CODASYL (INFORMATYKA 4/75),
- SZBD o cechach tzw. systemów samowystarczalnych *(self contained)*, których reprezentować może koncepcja E. F. Codd'a (INFORMATYKA 12/74), lub SZBD SYKON (INFORMATYKA 11/74),
- systemy wyszukiwania informacji niesformatowanej, reprezentantem których może być np. system STAIRS firmy IBM.

Te trzy klasy rozwiązań w zakresie gromadzenia, utrzymywania i wyszukiwania informacji nie wynikają wcale z faktu, że ich zastosowania są z punktu widzenia użytkownika rozdzielne. Wręcz odwrotnie, dostatecznie zaawansowany system informatyczny jakiejkolwiek organizacji będzie wymagał od aparatu obsługującego gospodarkę danymi cech charakterystycznych dla wszystkich trzech klas łącznie.
Docelowa koncepcja, o którą oparte są prace R-POLSYSTEMU w zakresie oprogramowania obsługującego gospodarkę danymi, dąży do integracji tych trzech klas rozwiązań — z punktu widzenia użytkownika.
Z punktu widzenia wewnętrznej organizacji baz danych i rozwiązań oprogramowania je obsługującego, różnice między poszczególnymi klasami są na tyle istotne, że nie wydaje się możliwe stworzenie takich rozwiązań z zakresu organizacji baz danych i takich metod dostępu, aby były dostatecznie efektywne jednocześnie dla wszystkich trzech — odpowiednich klasom rozwiązań — klas zastosowań. Dlatego też nie należy oczekiwać daleko idącej integracji z punktu widzenia oprogramowania, natomiast integrację z punktu widzenia otoczenia systemu gospodarki danymi uzyskać można dzięki „software'owej nadbudowie" rozpoznającej i rozdzielającej transakcje do obsługi przez odpowiednie „wyspecjalizowane" makro-moduły oprogramowania.
Pierwszy etap realizacji systemu gospodarki danymi zakłada stworzenie SZBD opartego o koncepcję Komitetu CODASYL i będącego pełną implementacją publikowanych w raportach z 1971 i 1973 języków DDL *(Data Description Language)* i DML *(Data Manipulation Language)*.

Pięć następujących podstawowych argumentów zaważyło na wyborze takiej koncepcji, realizowanego w pierwszej kolejności SZBD:

- jest to koncepcja ukierunkowana na obsługę baz danych w reżimie przetwarzania partiowego, a taki reżim przetwarzania będzie, naszym zdaniem, dominujący przez najbliższe lata, chociażby z uwagi na dostępny sprzęt JS. Nie oznacza to jednak, że gubimy z pola widzenia problemy obsługi baz danych na bieżąco
- jest to koncepcja ukierunkowana na obsługę baz danych w zastosowaniach przemysłowych, a przemysł to najbardziej liczący się potencjalny użytkownik. Ukierunkowanie koncepcji USZBD na zastosowania przemysłowe nie oznacza jednak, że nie pasuje ona do innych dziedzin zastosowań
- jest to koncepcja, która ma aktualnie największe szanse być podstawą przyjętych w przyszłości standardów w zakresie systemów zarządzania bazą danych
- jest to koncepcja, której głównym oponentem jest IBM. Nie ma więc podstaw by czekać na doświadczenia tej firmy w tym zakresie
- jest to koncepcja sprawdzona.

Koncepcja Komitetu CODASYL nie jest, być może, koncepcją spektakularną. Spowodowane jest to tym, że główny nacisk położono w niej nie na problem ułatwienia obsługi bazy danych, a na problem szeroko pojętej niezawodności bazy danych. I to jest chyba prawidłowe podejście.

Architektura Uniwersalnego Systemu Zarządzania Bazą Danych dla komputerów JS

Podstawowe elementy architektury USZBD i ich wzajemne relacje przedstawione są na rys. 1.
Omówmy pokrótce te elementy.

- Administrator bazy danych (ABD) jest to osoba (zespół ludzi) odpowiedzialna za zaprojektowanie i założenie bazy danych w czasie wdrażania systemu oraz za jej rozwój i bezpieczeństwo w okresie eksploatacji. Funkcje administratora bazy danych były przedmiotem oddzielnego artykułu (INFORMATYKA 1/75).

Administrator bazy danych komunikuje się z USZBD za pomocą przedstawionych na rys. 1 mechanizmów pośredniczących:
(1) — języka opisu danych (JOD)
(2) — języka opisu pamięci (JOP)
(3) — języka sterowania urządzeniami (JSU)

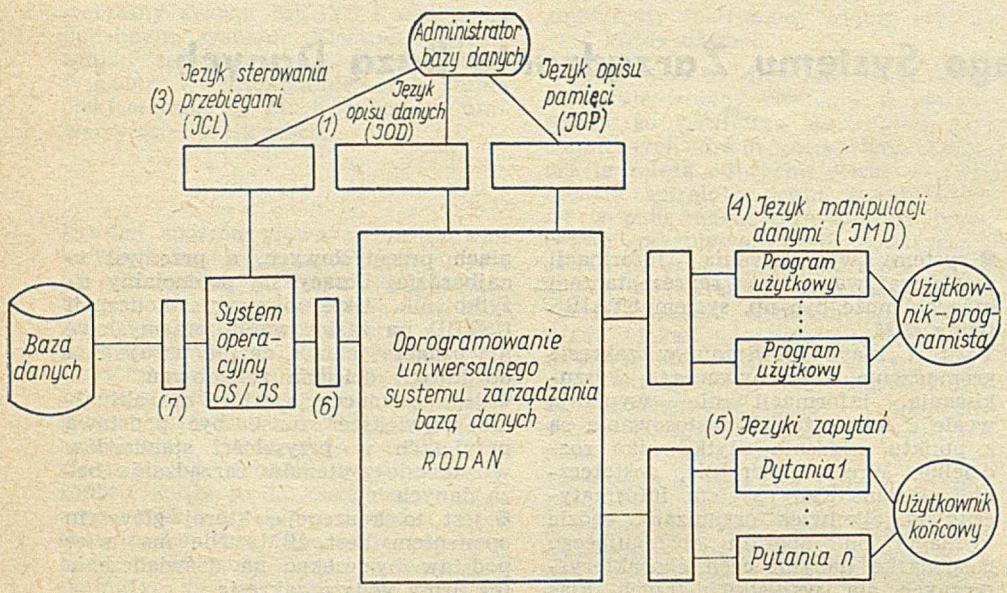
Język Opisu Danych jest aparatem definiowania logicznej struktury danych i oparty jest o opublikowany w Raporcie Komitetu CODASYL z 1973 roku język DDL *(Data Description Language)*.

Język Opisu Pamięci służy do definiowania fizycznej struktury bazy danych (struktury pamięci) i choć jest językiem oryginalnym, uwzględnia aktualne tendencje samego Komitetu CODASYL jak też i krytykę samego języka DDL.

Rolę **Języka Sterowania Urządzeniami** spełnia *Job Control Language* (JCL).

Mechanizmy opisu bazy danych będą przedmiotem następnego artykułu.
A w tej chwili chcielibyśmy zwrócić jedynie uwagę na fakt, że dostępny w USZBD aparat opisu bazy danych, poza samym zdefiniowaniem logicznej struktury danych i sposobu jej odwzorowania w pamięci, pozwala administratorowi bazy danych na włączanie własnych procedur obsługujących lub chociażby informujących go o stanie bazy danych i operacjach dokonywanych na bazie danych przez programy użytkownika.
Jest to naszym zdaniem aparat konieczny, aby administrator BD mógł spełniać swoje zadania.
W dotychczasowych znanych nam implementacjach języka DDL aparat ten (klauzula ON) był pomijany. Język JOD zawiera klauzulę ON syntaktycznie identyczną jak w DDL, przy czym intepretacja jej jest istotnie rozszerzona w takim kierunku, by udostępnić administratorowi bazy danych aparat pełnej kontroli nad operacjami dokonywanymi w bazie danych przez programy użytkownika.

● Użytkownik — programista jest programistą w dotychczasowym pojęciu, przy czym traktowany on jest z pewną dozą nieufności. Zakłada się, że programista (jak każdy człowiek) może robić błędy w swoich programach, natomiast baza danych nie może ulec zniszczeniu.
Administrator bazy danych, jako jedyny człowiek odpowiedzialny za stan i bezpieczeństwo bazy danych kontroluje działanie programów użytkownika i skutki tego działania — bez wiedzy programisty.
Programista komunikuje się z systemem za pośrednictwem Języka Manipulacji Danymi (JMD). Język ten jest niemal w pełni zgodny z opublikowanym w raporcie DBTG [1] Komitetu CODASYL z roku 1971, językiem DML *(Data Manipulation Language)*. Pełny opis języka JMD zostanie przedstawiony w lipcowym numerze INFORMATYKI.
JMD, jak i jego pierwowzór DML, nie jest językiem pełnym, a stanowi jedynie rozszerzenie tak zwanych języków bazowych *(host languages)*. W roli języków bazowych w pierwszej wersji USZBD wybrano PL/I i COBOL.



Wybór był raczej oczywisty i nie wymaga głębszych uzasadnień. W dalszej kolejności rozpatruje się rozszerzenie o JMD języka FORTRAN.
Język Manipulacji Danymi jest mocnym i elastycznym aparatem komunikacji programisty — użytkownika z bazą danych. Mimo to, nie należy zakładać, że praca programisty będzie istotnie ułatwiona w stosunku do przetwarzania tradycyjnych (niepowiązanych) plików w takich językach jak PL/I czy COBOL. Wystarczającym jednak osiągnięciem jest to, że praca programisty dzięki JMD nie będzie utrudniona wskutek przejścia od przetwarzania oddzielnych plików danych do przetwarzania baz danych o dowolnie skomplikowanej logicznej i fizycznej organizacji.

● Użytkownik końcowy. Tym terminem określa się użytkownika — nieprogramistę. Komunikuje się on z systemem za pośrednictwem Języka Zapytań (JZ). JZ musi być z założenia prosty, łatwy do nauczenia i przystosowany do problematyki konkretnego zastosowania. Ponadto, należy również zwrócić uwagę na fakt jak istotny wpływ na sposób konwersji ma rodzaj urządzeń końcowych. Zupełnie inne możliwości daje np. końcówka typu elektryczna maszyna do pisania, a inne końcówka typu monitor ekranowy z piórem świetnym.
Trzecim elementem, który ma wpływ na postać języka zapytań jest organizacja bazy danych. Inny jest język dla baz danych opartych o zbiory inwersyjne, inny przy relacyjnym modelu bazy danych i inny wreszcie przy sieciowej organizacji bazy danych.
Tak więc język zapytań jest silnie uzależniony od zastosowania modelu bazy danych i od urządzeń końcowych. Stąd też rodzi się uzasadniona wątpliwość w możliwość stworzenia uniwersalnego języka zapytań, który byłby łatwy czyli przystosowany do zastosowania, wykorzystywał możliwości różnych urządzeń końcowych i w sposób optymalny chodziłby po strukturach danych. Uwzględniając ten fakt koncepcja USZBD zakłada, prócz wzorcowego języka zapytań ukierunkowanego na najbardziej typowe zastosowania, wybrane urządzenia końcowe i standardową organizację bazy danych, stworzenie aparatu umożliwiającego użytkownikowi definiowanie jego własnych języków.
Idea stworzenia aparatu umożliwiającego użytkownikowi definiowanie jego własnych języków zapytań nie jest ideą nową. Nie jest jednak jeszcze w chwili obecnej rzeczą łatwą stworzenie aparatu umożliwiającego realizację tej idei w praktyce. Problemowi temu poświęcony będzie oddzielny artykuł.

● Oprogramowanie Uniwersalnego Systemu Zarządzania Bazą Danych jest zbiorem produktów programowych zabezpieczających obsługę bazy danych lub dostęp do bazy danych zgodnie z poleceniami przekazywanymi przez:
— administratora bazy danych za pośrednictwem języków JOD i/lub JOP
— programistę — użytkownika w formie zdań języka JMD
— użytkownika końcowego w formie zdań języka JZ.

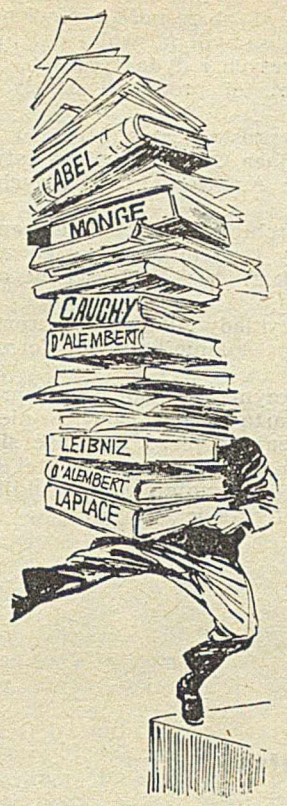
Oprogramowanie USZBD zapewnia w pierwszym etapie obsługę pracy wieloprogramowej (równoległe przebiegi w różnych strefach pamięci) w trybie przetwarzania wsadowego, a następnie w miarę pojawiania się odpowiedniego sprzętu JS, rozbudowywany będzie o obsługę pracy wielozadaniowej w trybie przetwarzania zdalnego.
Z punktu widzenia konstrukcji, Oprogramowanie USZBD jest zbiorem: translatorów, generatorów, parametryzowanych procedur i makroinstrukcji. Struktura oprogramowania USZBD omówiona zostanie w kolejnym artykule cyklu.

● System Operacyjny. Oprogramowanie USZBD realizowane jest na komputery JS z systemem operacyjnym OS/JS.

● Baza danych. Organizacja bazy danych jest definiowana przez ABD zarówno na poziomie struktury danych jak i struktury pamięci.
W odróżnieniu od dotychczasowych implementacji CODASYL'owskiej koncepcji USZBD, poprzez wprowadzenie języka opisu pamięci dopuszcza różne organizacje struktury pamięci. Co więcej, w ramach jednej bazy danych poszczególne obszary mogą mieć różne organizacje.
Standardowa organizacja pamięci (tj. organizacja przyjmowana przez system przez domniemanie, w przypadku gdy ABD nie poda opisu pamięci) jest organizacją maksymalnie elastyczną dopuszczającą zapisy (rekordy) o zmiennej i teoretycznie nieograniczonej długości.
Zbiory *(sets)* są odwzorowywane w formie łańcuchów lub tablic adresowych.
Bardziej szczegółowe informacje o organizacji bazy danych podane będą przy okazji omawiania języków opisu bazy danych w następnym artykule tego cyklu.

Władysław Bogucki
Witold Staniszkis

[1] *Data Base Task Group.*



Recenzja książki Janusza Hawryluka: Maszyna cyfrowa—narzędzie człowieka współczesnego

Mimo stosunkowo dużej liczby polskich publikacji książkowych i obcych tłumaczeń poświęconych informatyce i ukazujących się na krajowym rynku wydawniczym wciąż odczuwa się zapotrzebowanie na pozycje przeznaczone dla szerokiego grona ludzi stosujących informatykę w swej pracy zawodowej w sposób pomocniczy i uzupełniający oraz dla ludzi, którzy dopiero rozpoczęli zajmować się informatyką zawodowo.

To utrzymujące się zapotrzebowanie wynika — moim zdaniem — z faktu, że większość książek z tej dziedziny często dotyczy bądź zagadnień i spraw bardzo specjalistycznych, bądź pisana jest w sposób tak trudny i zawiły, że aż odstraszający potencjalnych Czytelników z poza kręgu „wtajemniczonych" w problemy informatyki.

Tak się szczęśliwie złożyło, że niebezpieczeństw tych udało się uniknąć Januszowi Hawrylukowi w jego książce „Maszyna cyfrowa — narzędziem człowieka współczesnego", która doskonale odpowiada wspomnianemu wyżej zapotrzebowaniu. Książka Janusza Hawryluka charakteryzuje się problemowo rozległym omówieniem przedmiotu, jakim we współczesnym życiu społecznym, naukowym i gospodarczym stała się maszyna cyfrowa.

Autor rozpatruje problemy, jakie wnosi sobą maszyna cyfrowa, z punktu widzenia — jak sam mówi we wstępie — pięciu poziomów, na które składają się takie zagadnienia jak: elektronika, logika, programowanie, eksploatacja i zastosowania.

Ponieważ książka jest przeznaczona nie dla konstruktorów a dla szerokiego grona odbiorców dwa pierwsze poziomy potraktowane zostały bardziej ogólnie, jednakże całkowicie wystarczająco, aby zrozumieć istotę działania maszyny cyfrowej.

Układ tematyczny kolejnych rozdziałów książki pozwala Czytelnikom prześledzić dostatecznie szeroki wachlarz tematów charakteryzujących niemal całą dziedzinę informatyki — począwszy od ogólnych rozważań na temat informacji i jej przetwarzania — a skończywszy na zastosowaniach maszyn cyfrowych oraz przewidywaniach obejmujących rozwój techniczny i perspektywy gospodarcze.

Poza wstępem, rozdziały 2 i 3 poświęcone zostały omówieniu podstawowych pojęć, takich jak: informacja, przetwarzanie informacji, ilość informacji itp. w odniesieniu do dyscypliny naukowej za jaką uznana została teoria informacji. Omówione zostały tutaj również trzy grupy problemów związanych z pojęciem informacji:

- problem techniczny (sposób przenoszenia informacji — sygnały)
- problem semantyczny (kwestia umowności znaczeń przypisywanych sygnałom — kody)
- problem pragmatyczny (wpływ informacji na podejmowanie decyzji).

Rozdział 4 poświęcony jest całkowicie opisowi struktury i zasad działania maszyny cyfrowej. Czytelnik zapoznany zostaje ze strukturą maszyny cyfrowej, charakterystyką i przeznaczeniem jej bloków funkcjonalnych, z takimi pojęciami jak rozkaz, program i algorytm oraz poznaje sposób w jaki maszyna cyfrowa realizuje program.

Wyposażenie techniczne maszyn cyfrowych oraz istota programowania zostały podane w rozdziałach 5 i 6.

Autor charakteryzuje moduły sprzętowe zestawu maszyny cyfrowej tj.: procesor, pamięć, urządzenia wejścia i wyjścia, urządzenia transmisji i konwersji danych. Podaje zasady i działania oraz omawia ich rolę i udział w procesie przetwarzania. Podane i omówione zostały przykłady zestawów maszyn cyfrowych w relacji do ich mocy obliczeniowej oraz typowych zastosowań.

Rozdział 6 omawia programowanie maszyn cyfrowych.

Autor — to jest moim zdaniem dodatkową dydaktyczną wartością pracy — nie tworzy, jak to często można spotkać w innych książkach, fikcyjnego języka programowania, ale dla omówienia tajników programowania korzysta z wybranych elementów języka ALGOL 60, podając jednocześnie przykłady programów napisanych w wersji ICL tego języka na maszynie ODRA 1304.

*) WNT, Warszawa 1974, wyd. I, s. 434, ark. wyd. 22.

Rozdział 7 poświęcony jest opisowi oprogramowania podstawowego i użytkowego, zapewniającego właściwą i wydajną pracę całego zestawu maszyny cyfrowej. Scharakteryzowane zostały w tym rozdziale spotykane aktualnie metody pracy maszyn cyfrowych: praca wsadowa, praca wieloprogramowa, wieloprocesorowość, praca bezpośrednia, przetwarzanie w czasie rzeczywistym, wielodostępnym i praca w trybie konwersacyjnym.

Szeroki pogląd z dziedziny zastosowań maszyn cyfrowych podaje Autor w rozdziale 8.

Poczynając od badań naukowych — poprzez zarządzanie, sterowanie procesami, symulację, projektowanie, zastosowania administracyjne, medyczne, programowanie, nauczanie — a skończywszy na tak frapujących przykładach, jak np.: zastosowanie maszyny cyfrowej do badania okoliczności zabójstwa prezydenta Kennedy'ego.

Książkę kończą rozważania Autora na temat perspektyw rozwoju współpracy człowieka z maszyną, przytoczone w rozdziale 9.

Mam nadzieję, że przytoczona powyżej tematyczna charakterystyka książki Janusza Hawryluka „Maszyna cyfrowa — narzędzie człowieka współczesnego" uzmysławia przyszłemu Czytelnikowi obszerność omówionych w książce problemów i tym samym pozwala zorientować się w jej walorach poznawczych i dydaktycznych.

Z innych zalet tej produkcji należy moim zdaniem wymienić:

- wyjątkowo prostą, przystępną i komunikatywną formę opisu pozwalającą Czytelnikowi na łatwe zrozumienie omawianych przez Autora często trudnych i skomplikowanych zagadnień
- właściwy, problemowy układ sekwencji rozdziałów książki pozwalający Czytelnikowi na ich wybór zgodnie z jego zainteresowaniami bez trudności w odbiorze prezentowanego materiału.
- czytelne przykłady ułatwiające zrozumienie opisywanych zagadnień, oraz komunikatywna forma ilustracji graficznych.

Wszystko to czyni — moim zdaniem — książkę Janusza Hawryluka wartą polecenia i to nie tylko Czytelnikom, którzy zamierzają dopiero zapoznać się z informatyką, ale również i tym, którzy pracują w tej dziedzinie zawodowo. Z pewnością i oni znajdą w tej publikacji, wiele interesujących ich materiałów.

Józef Czajkowski

Bibliografia wydawnictw polskich z dziedziny informatyki

● **Automatyzacja procesów technologicznych w przemyśle maszynowym** — TYMOWSKI J., Wyd. 2 całkowicie przerobione, WNT, Warszawa 1975, ss. 271, cena zł. 35.

Wiadomości podstawowe. Kierunki automatyzacji. Automatyzacja prostych cyklów pracy. Automatyzacja zasilania obrabiarek. Automatyzacja transportu międzystanowiskowego. Automatyzacja kontroli. Sterowanie programowe. Sterowanie numeryczne. Stosowanie obrabiarek sterowanych numerycznie. Zastosowanie maszyn matematycznych w technicznym przygotowaniu produkcji. Automatyzacja montażu. Ocena ekonomicznej efektywności automatyzacji. Postępowanie przy pracach nad automatyzacją. Czynniki wpływające na tempo automatyzacji.
Książka przeznaczona jest dla inżynierów pracujących w przemyśle maszynowym oraz dla studentów wydziałów mechanicznych wyższych szkół technicznych.

● **Automatyczne przetwarzanie danych. System 360** — BROOKS F., IVERSONK. Tłum. wyd. ang. z roku 1969. WNT, Warszawa 1975, ss. 434, cena zł 60.

Podstawy przetwarzania danych. Urządzenia do ręcznego przetwarzania danych. Maszyny licząco-analityczne. Kodowanie programów. Organizacja maszyn cyfrowych na przykładzie maszyny IBM 360/30. Programowanie. Wyszukiwanie i sortowanie. Systemy programowania. Projektowanie systemów przetwarzania danych.
Książka przeznaczona jest dla programistów i projektantów systemów EPD oraz studentów kierunków informatycznych.

● **Programowanie symulacji procesów** — WINKOWSKI J., WNT, Warszawa 1974, ss. 207, cena zł 22.

Algorytmy symulacji. Symulacja procesów złożonych. Programowanie symulacji. Typowe zastosowania. Symulacja procesów współdziałania. Systemy algorytmów i algorytmy złożone. Dodatki: Pojęcia ogólnomatematyczne. Algorytmy. Podstawowe pojęcia teorii prawdopodobieństwa. Generacja miar prawdopodobieństwa.
Książka przeznaczona jest dla osób zajmujących się zastosowaniami ETO a w szczególności dla projektantów języków i systemów do programowania symulacji.

● **Symulacja systemów** — GORDON G., Tłum. wyd. ang. z roku 1969. WNT, Warszawa 1974, ss. 339, cena zł 45.

Modele systemów. Symulacja systemów. Symulacja systemów ciągłych. Dynamiczne modele przemysłu. Programowanie w języku DYNAMO. Elementy rachunku prawdopodobieństwa. Strumienie zgłoszeń i czasu obsługi. Symulacja systemów dyskretnych. Symulacja przy użyciu języka FORTRAN.
Technika pisania programów symulacyjnych. Wprowadzenie do języka GPSS. Przykład zastosowań języka GPSS. Wprowadzenie do języka SIMSCRIPT. Organizacja zbiorów w języku SIMSCRIPT. Weryfikacja wyników symulacji.
Książka przeznaczona jest dla projektantów i analityków systemów oraz studentów wyższych uczelni technicznych.

● **Problemy przetwarzania informacji T. 2.** — MAZURKIEWICZ A. — red., WNT, Warszawa 1974, ss. 247, cena zł 35.
O przyjęciu maszyny liczącej — PAWLAK Z.
Podstawy teorii programowania — MAZURKIEWICZ A.
Wybrane zagadnienia lingwistyki matematycznej — BLIKLE A.
Niektóre zagadnienia struktur danych — TURSKI W. M.
Symulacja algorytmiczna — WINKOWSKI J.
Książka zaznajamia czytelników z polskimi pracami z dziedziny matematycznych podstaw informatyki. Książka przeznaczona jest dla projektantów języków i systemów programowania oraz studentów wyższych lat wydziałów matematycznych.

● **Człowiek w świecie informacji** — KOWALCZYK E., Książka i Wiedza, Warszawa 1974, ss. 285, cena zł 25.
Prognozy — Perspektywy
Podstawy pojęcia informacji. Informacja a system nerwowy. Język a informacja. Informacja a myślenie. Zjawiska i procesy informacyjne w nauczaniu. Informacja w organizacji nauki i badań naukowych. Informacja a propaganda. Niektóre aspekty socjologiczne problemów. Informacja i środki łączności w organizacji, zarządzaniu i gospodarce.
Zarys rozwoju techniki przekazywania informacji na odległość. Komputery i komputeryzacja: historia komputerów, ogólna charakterystyka komputerów, projektowanie systemów informatycznych, zastosowania komputerów, banki danych, komputery czwartej generacji.
Książka stanowi ogólne popularne wprowadzenie do współczesnej wiedzy o problemach informacyjnych.

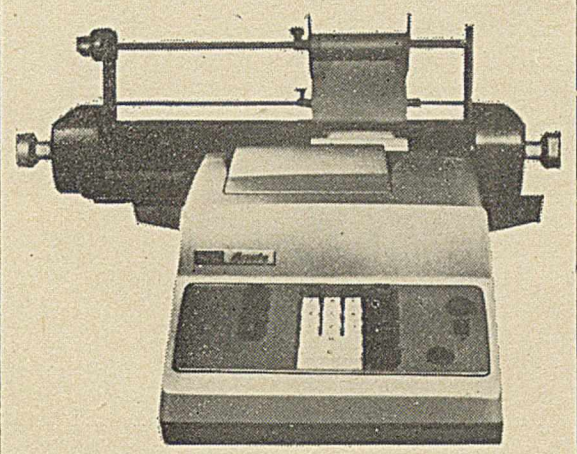
● **Projektowanie systemów elektronicznego przetwarzania danych** — BOGDANOWICZ Z., KOLBUSZ E., KRAM E., Wyd. Ośrodka Badawczo-Rozwojowego Informatyki INFORNA, Warszawa 1974, ss. 204, cena zł 58.
Problemy informatyki.
Podstawowe pojęcia przetwarzania sekwencyjnego. Zasady komputerowej technologii przetwarzania danych. Projektowanie systemu informacyjnego. Projektowanie technologii przetwarzania. Niektóre problemy organizacji projektowania i wdrażania systemów EPD.
Materiały przeznaczone są dla projektantów systemów EPD.

bme

Büromaschinen-Export GmbH
Berlin, DDR — 108 Berlin, Friedrichstrasse 61, Niemiecka Republika Demokratyczna

daro 1360 – OCR
albo: jak usprawnić bieżące kompleksowe prowadzenie rozliczeń w przedsiębiorstwie.

daro 1360 — OCR jest drukarką pisma dostosowanego do odczytu optycznego. Zapis odbywa się na taśmach rejestracyjnych czcionką o znormalizowanym kroju. Zapis ten jest czytelny zarówno dla człowieka, jak i dostosowany do odczytu automatycznego przez czytnik dokumentów.
Po prostu: maszynowym nośnikiem danych jest sam dokument źródłowy, który tym samym może być bezpośrednio wczytywany do komputera.
A więc rozwiązanie najwłaściwsze w tych systemach przygotowania i przetwarzania danych, gdzie wymagane są szybkość i niezawodność.
Unikacie czasochłonnego pośrednictwa taśm lub kart dziurkowanych, czy taśm magnetycznych!
Dzięki drukarce daro 1360 —OCR zmniejszacie pracochłonność w statystyce lub księgowości, przyspieszacie i zwiększacie możliwość wykorzystania Waszych operacji rozliczeniowych.



daro

1360

Żądajcie bliższych informacji.
Służymy naszymi materiałami informacyjnymi.
Przedstawicielstwo BME w Polsce,
Biuro Techniczno-Handlowe przy Ambasadzie NRD
Warszawa, ul. Filtrowa 62 m 63
Sprzedaż i informacje
Predom-Org.
00-033 Warszawa,
ul. Górskiego 9

Prosimy odwiedzić nas na Międzynarodowych Targach Technicznych w Poznaniu od 8.VI. do 17.VI.1975.

WCT/431/K/75-B

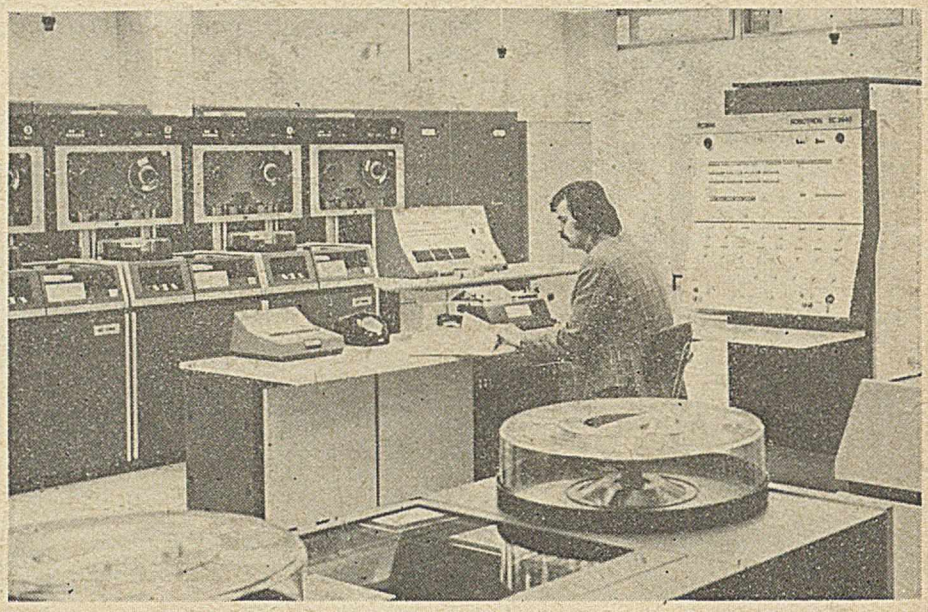
robotron

Problemy
Wymagania
Rozwiązania

Stają przed Wami podobnie, jak przed nami!

Nasza droga:

JS — Jednolity System Elektronicznych Maszyn Cyfrowych.
To symbol współpracy nie znającej granic państwowych — olbrzymi potencjał w służbie producentów i użytkowników.



Nasza oferta:

Komputer do przetwarzania danych ROBOTRON EC 1040 ze zmienną liczbą urządzeń zewnętrznych JS do rozwiązywania zadań gospodarczych i naukowo-technicznych, do zwiększania efektywności zarządzania, planowania i rozliczeń, do zastosowania jako komputer główny w systemie hierarchicznym. Oprogramowanie systemu ukierunkowane maszynowo i problemowo, szkolenie dostosowane do konkretnych wymagań użytkownika, wypróbowany serwis techniczny — to zakres świadczonych przez nas usług!

Program dostosowany do Waszych potrzeb! Korzystajcie z bardziej szczegółowych informacji!

WCT/431/K/75-A

bme

Eksporter:

Büromaschinen-Export GmbH Berlin
DDR — 108 Berlin,
Friedrichstrasse 61
Niemiecka Republika Demokratyczna
Przedstawicielstwo BME w Polsce
Biuro Techniczno-Handlowe przy Ambasadzie NRD
Warszawa,
ul. Filtrowa 62 m. 63

Sprzedaż i informacje:
Biuro Generalnych Dostaw MERA-ELWRO-SERVICE
Wrocław, ul. Ostrowskiego 32